NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SYNCHRONIZATION ANALYSIS AND SIMULATION OF A STANDARD IEEE 802.11G OFDM SIGNAL

by

Keith D. Lowham

March 2004

Thesis Advisor:	Frank E. Kragh
Second Reader:	R. Clark Robertson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Synchronization Analysis and Simulation of a Standard IEEE 802.11g OFDM Signal			**5. FUNDING NUMBERS**
6. AUTHOR(S) Lowham, Keith D.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Synchronization of orthogonal frequency-division multiplexed (OFDM) signals is significantly more difficult than synchronization of a single-carrier system. The recently approved IEEE Standard 802.11g specifies a packet-based OFDM system that provides a basis for the discussion of OFDM synchronization in a packet-based environment. Algorithms that synchronize the receiver carrier demodulation frequency and phase, the data frame, the OFDM symbol timing, and the data symbol timing are discussed and analyzed in an AWGN channel. System View simulation is used to implement the frame and carrier frequency synchronization algorithms, where the performance of these algorithms is analyzed and they are shown to be useful detection algorithms for Standard 802.11g signal reception.

14. SUBJECT TERMS OFDM, Orthogonal Frequency Division Multiplexing, 802.11, 802.11g, AWGN, PBCC, DSSS, DSSS-OFDM, Synchronization, Carrier Synchronization, Frequency Synchronization, Phase Synchronization, Symbol Synchronization, System View, FFT, IFFT, Cyclic Prefix, Guard Interval, wireless LAN			**15. NUMBER OF PAGES** 176
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SYNCHRONIZATION ANALYSIS AND SIMULATION OF A STANDARD IEEE 802.11G OFDM SIGNAL

Keith D. Lowham
Lieutenant Commander, United States Navy
B.S., California State University, Chico, 1986

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2004

Author: Keith D. Lowham

Approved by: Frank E. Kragh
Thesis Advisor

R. Clark Robertson
Second Reader

John P. Powers
Chairman, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Synchronization of orthogonal frequency-division multiplexed (OFDM) signals is significantly more difficult than synchronization of a single-carrier system. The recently approved IEEE Standard 802.11g specifies a packet-based OFDM system that provides a basis for the discussion of OFDM synchronization in a packet-based environment. Algorithms that synchronize the receiver carrier demodulation frequency and phase, the data frame, the OFDM symbol timing, and the data symbol timing are discussed and analyzed in an AWGN channel. System View simulation is used to implement the frame and carrier frequency synchronization algorithms, where the performance of these algorithms is analyzed and they are shown to be useful detection algorithms for Standard 802.11g signal reception.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

"Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity."

General George S. Patton

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The recent approval of the IEEE Standard 802.11g has increased the use of orthogonal frequency-division multiplexing (OFDM) in devices that are in wide use. Coupled with the rapid increase in the use of wireless networking, this widespread technology has become a major component of wireless devices not only in the consumer electronics industry, but it is also widely used by the governments and the military of many countries. With this increase in implementation comes a concern with the ability to receive a useable signal in the most challenging environments. This means not only in a consumer electronics environment that is frequently crowded with other wireless devices operating in the same frequency spectrum, but also in military environments that may include the use of jamming and interception by a hostile force.

In this thesis, the IEEE Standard 802.11g is closely examined. Physical layer aspects of the standard are discussed along with how they interface with the previous IEEE 802.11 standards. Implementation of a receiver and transmitter using the OFDM mode of Standard 802.11g is detailed and then used in implementing and testing synchronization algorithms.

There are many aspects to synchronizing the receiver in an OFDM system with the transmitted signal. In addition, in a packet-based communications system such as Standard 802.11g compliant systems, the receiver must synchronize with the transmitter very quickly. Even with the packet length using the maximum allowed number of data bytes, the entire packet can be at most 627 μs long at a data rate of 54 Mbps. The preamble and the header are a total of 20 μs long, leaving little time to achieve synchronization before the start of the data. Since the header contains information required for proper demodulation of the OFDM signal, synchronization must be accomplished in the time allotted for the long and short training sequences, which is 16 μs. Once synchronization is achieved, it must be maintained over the duration of the OFDM symbol. If the channel

characteristics change significantly during the transmission of the packet, synchronization and the data can be lost, resulting in retransmission and an overall slower data reception rate.

Finally, frame and carrier frequency synchronization circuits are built using the System View simulation software and compared with theoretical expectations.

I. INTRODUCTION

The recent approval of the IEEE Standard 802.11g has increased the use of orthogonal frequency-division multiplexing (OFDM) in devices that are in wide use. Coupled with the rapid increase in the use of wireless networking, this widespread technology has become a major component of wireless devices not only in the consumer electronics industry, but it is also widely used by the governments and the military of many countries. With this increase in implementation comes a concern with the ability to receive a useable signal in the most challenging environments. This means not only in a consumer electronics environment that is frequently crowded with other wireless devices operating in the same frequency spectrum, but also in military environments that may include the use of jamming and interception by a hostile force.

One aspect of optimizing the reception of any signal is the synchronization of the receiver with the transmitter. There are several different types of synchronization needed to ensure the receiver correctly decodes the transmitted signal. This thesis describes the synchronization process used in OFDM systems and algorithms that optimize the synchronization in the presence of additive white Gaussian noise (AWGN). The synchronization algorithms are then implemented using System View simulation software by Elanix Inc.

A. SCOPE

OFDM has been implemented in several standards, including IEEE Standards 802.11a and 802.11g, HIPERLAN II, and Digital Video Broadcast (DVB) to name only a few. The optimum methods of synchronization differ, depending on the type of signal being transmitted. If the signal is being continuously transmitted, as in DVB, a receiver has a longer time to obtain synchronization, and the addition of dedicated blocks for synchronization only serve to slow down the data transmission. In a packet-based system such as IEEE Standard 802.11g and HIPERLAN II, the receiver must quickly obtain synchronization to be ready for the short bursts of data that occur at random intervals. This requires the use of quick, synchronization-block-based systems that can obtain synchronization

quickly at the expense of using some of the packet for the special synchronization blocks. This thesis focuses on the synchronization issues surrounding the packet-based IEEE Standard 802.11g.

1. Introduction to OFDM

OFDM is a multi-carrier frequency modulation scheme in which the carrier frequencies used are orthogonal to each other. This means that all the frequencies are uncorrelated and that the signals modulated by these orthogonal frequencies can be recovered by correlating the received signal with each of the original carrier frequencies. Two signals $x(t)$ and $y(t)$ are defined as being orthogonal when

$$\int_{t}^{t+T} x(s)y(s)\,ds = 0 \tag{1.1}$$

where T is the period of the signals. If the transmitted signal is proportional to one of the orthogonal signals, $Ax(t)$, for example, then the modulated signal A can be recovered by correlating the received signal with both $x(t)$ and $y(t)$ and choosing the output with the largest value. If the received signal is correlated with $y(t)$ then

$$r = \int_{t}^{t+T} Ax(s)y(s)\,ds \tag{1.2}$$

But from (1.1) the integral of (1.2) is zero. However, if the received signal is correlated with $x(t)$ then the output is $A\int_{t}^{t+T} x^2(s)\,ds,$ the desired signal. This is the basis for OFDM. Although the correlation can be used with all multi-carrier systems, the orthogonality of the signals in OFDM allows the transmitted signals to overlap in the frequency-domain, reducing the required bandwidth for transmission. For example, the required null-to-null bandwidth for a binary phase-shift keying (BPSK) signal using a data rate of 6 Mbps and a rate-$1/2$ encoding scheme requiring a total data rate of 12 Mbps uses a 24-MHz bandwidth. However, this same signal transmitted using the IEEE Standard 802.11a or 802.11g OFDM modulation uses only 16.6 MHz of bandwidth. This savings in bandwidth allows the transmission of higher data rates over the same bandwidth as the lower

frequency data that use different modulation techniques, and is a major factor in the growth of OFDM based systems.

OFDM was originally conceived by Chang [1] in 1966. Because of the difficulty in using a bank of modulators to produce a signal that takes full advantage of the orthogonality of the sub-carriers, this concept was not implemented in real systems until the idea of using a discrete Fourier transform to modulate the signals instead of using a bank of modulators was proposed by Weinstein [2] in 1971. This technique was quickly adapted to the use of the fast Fourier transform (FFT) to make modern OFDM systems practical.

2. Interference in OFDM

Interference in high data rate systems revolves around the overlap of data symbols that are adjacent in frequency or consecutive in time, which only gets worse as the data rate increases. This occurs because of the channel characteristics between the transmitter and the receiver, with multipath propagation being one of the major causes of this interference. This interference is called intersymbol interference when the signal overlap occurs in the time-domain, and it is call intercarrier interference when the interference occurs in the frequency-domain. One of the big advantages of OFDM over other modulation techniques is its reduction in intersymbol interference seen in a multipath environment. Since the transmitted data can be divided up and modulated onto several different carriers, the data rate for each carrier is reduced. This lower data rate reduces the intersymbol interference in the multipath channel.

An example of intersymbol interference is shown in Figure 1. The transmitted signal is a sharp square wave, but the channel has distorted the received signal. This has caused some of the signal from adjacent pulses to leak into the expected time period of the next pulse. This overlap is called intersymbol interference and can be corrected by increasing the separation between the pulses enough so that the received signals do not overlap, or correspondingly reducing the data rate so more of the received signal is unaffected by adjacent symbols. This separation with no data between pulses or signals is called a guard interval.

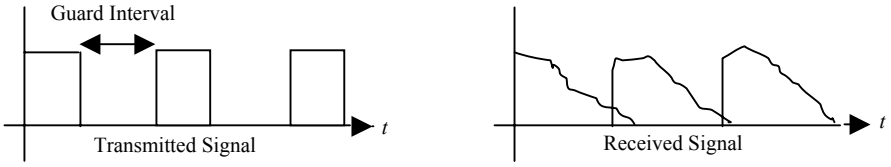

Figure 1 Intersymbol Interference.

The OFDM symbol uses a guard interval to reduce the effects of intersymbol inference between adjacent OFDM symbols, and it uses a cyclic extension of the OFDM symbol to reduce intercarrier interference. Since the Fourier transform of a square wave in the time-domain is a *sinc* function in the frequency-domain, an abrupt cutoff of the signal in the time-domain causes spectral spreading. The purpose of the guard interval and cyclic extension of the transmitted signal is to separate OFDM symbols enough to prevent intersymbol interference and to avoid the sharp end to the time-domain signal in order to prevent intercarrier interference.

3. Synchronization of OFDM Signals

For a packet-based OFDM transmission system, the synchronization of the signals can be broken down into five different steps. These are frame synchronization, carrier frequency synchronization, carrier phase synchronization, OFDM symbol synchronization, and data symbol synchronization. The frame synchronization involves detecting the presence of an incoming packet of information. For a packet transmission system, this must be done quickly to ensure there is enough time to obtain the other required synchronizations before the data is transmitted. The carrier synchronization has been split into two steps, since typically the frequency is synchronized with a phase-locked loop configuration, which does not synchronize the phase of the transmitted and received signals. Once the frequencies have been synchronized then the phases can be synchronized to optimize the data reception. Finally, the timing of the OFDM symbols and the data symbols is required to properly demodulate the received data.

The process of synchronization in an OFDM system differs from the process of synchronization in a single-carrier system in that OFDM signals have many frequencies that can potentially interfere with each other if the orthogonality of the signals is lost. In a single-carrier system it is relatively easy to separate out the signal of interest, filter it, and compare it to the internally generated clock to achieve a synchronized system. Doing this with an OFDM signal simply produces several sub-carrier frequencies that are close enough together that filtering one signal of interest is impractical. The goal with OFDM systems is to process the received signal in such a way as to produce a single value that is proportional to the timing error and use this error signal to drive a voltage-controlled oscillator. This is similar to the single-carrier system, but the algorithms used to produce the error signal can be quite complex, as will be shown in Chapter IV.

This introduction to OFDM has provided only a cursory description of OFDM and the many issues involved with transmitting and receiving these signals. However, References [3] through [7] provide excellent discussions and descriptions of OFDM signaling and are recommended reading for the interested reader needing more details on the basics of OFDM.

B. THESIS ORGANIZATION

This thesis is organized into an introduction chapter, a conclusion chapter and four other chapters. Chapter II describes the OFDM packet-based system using the IEEE Standard 802.11g with a discussion of the other 802.11 standards and how they affect the design of the Standard 802.11g. Chapter III implements an 802.11g OFDM transmitter and receiver using the System View simulation software by Elanix. This simulation allows for the analysis of the synchronization algorithms and the possible incorporation of these designs into a Field Programmable Gate Array (FPGA) design for practical implementation of a real system. Chapter IV introduces the various algorithms available to synchronize an OFDM system and performs an analysis of the packet-based frequency and packet synchronization algorithms in the presence of AWGN. Chapter V takes the algorithms developed in Chapter IV and implements them using System View. Finally, this thesis concludes with an overview of the results and a recommendation for further

study. Appendix A show the file format for a System View token input, and Appendix B shows the Mathcad 2001 programming used in this thesis.

II. IEEE STANDARD 802.11G DESCRIPTION

This chapter provides a description of the IEEE Standard 802.11g. Since the latest of the 802.11 standards is based on the previous standards in the 802.11 family, a brief description of the other standards is also given to form an understanding of how Standard 802.11g is formatted.

A. IEEE STANDARD 802.11

The original wireless local area network (LAN) physical and medium access control layers standard by IEEE was Standard 802.11-1997 [8], established in June 1997 as a method to provide wireless connectivity to automatic machinery, equipment, or stations that required rapid deployment, including portable or hand-held devices, or devices that could be mounted on moving vehicles within a local area. The data rates supported by the original standard were 1 Mbps and 2 Mbps using frequency-hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS), and infrared transmission with the 2-Mbps data rate optional when using infrared or FHSS.

The modulation scheme used in the Standard 802.11-1997 FHSS standard includes binary Gaussian frequency-shift keying (2-GFSK) which uses Gaussian pulse shaping on a binary frequency-shift keying (BFSK) signal to achieve the 1-Mbps data rate. The 2-Mbps data rate is achieved by using a 4-GFSK-modulation scheme. In the DSSS case, differential binary phase-shift keying (DBPSK) is used to transmit the 1-Mbps data rate and differential quadrature phase-shift keying (DQPSK) is used to transmit at 2 Mbps. For infrared transmission, the 1-Mbps data rate uses a sixteen-position pulse position modulation (16-PPM) in which four data bits are mapped into one position of a 16-PPM symbol. The 2-Mbps data rate encodes two data bits into a symbol and then maps these into one position of a 4-PPM symbol. [8]

In September of 1999, the data rate of the basic standard was extended with IEEE Standard 802.11b [9], which supports data rates of 5.5 Mbps and 11 Mbps using direct sequence spread spectrum and complementary code keying (CCK) to extend the data rate but not increase the occupied bandwidth. Optional capabilities in this standard include the

use of frequency agility and packet binary convolutional coding (PBCC). Frequency-hopping spread spectrum and infrared transmission were not included in this update of the standard. This high rate extension allows for backwards compatibility with the earlier standard by keeping the same physical layer preamble and header format so both physical layers can co-exist in the same basic service set.

Complementary code keying is a modulation scheme using a chipping sequence that is determined by the incoming data vice a predetermined pseudorandom sequence, as is performed in traditional direct sequence spread spectrum. In the Standard 802.11-1997, a fixed 11-chip pseudorandom Barker sequence was used to spread the data. The Standard 802.11b uses an 8-chip CCK sequence defined as

$$c = \left\{ e^{j(\varphi_1+\varphi_2+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_2+\varphi_4)}, -e^{j(\varphi_1+\varphi_4)}, e^{j(\varphi_1+\varphi_2+\varphi_3)}, e^{j(\varphi_1+\varphi_3)}, -e^{j(\varphi_1+\varphi_3)}, e^{j\varphi_1} \right\} \quad (2.1)$$

where φ_1 is the phase differential determined by the first two bits (d_0 and d_1) of the four bits used in the CCK symbol for the 5.5 Mbps data rate and is defined in Table 1. The remaining phases in (2.1) are given by $\varphi_2 = \pi d_2 + (\pi/2)$, $\varphi_3 = 0$, and $\varphi_4 = \pi d_3$. [9]

Table 1 Complementary Code Keying DQPSK [From Reference 9.]

Bits d_0 and d_1	**Even Symbol Phase change**	**Odd Symbol Phase Change**
00	0	π
01	π/2	3π/2
11	π	0
10	3π/2	π/2

For the all-zero data sequence, the chipping sequence becomes

$$c = e^{j\varphi_1}\left\{e^{j\left(\frac{\pi}{2}\right)}, e^{j(0)}, e^{j\left(\frac{\pi}{2}\right)}, -e^{j(0)}, e^{j\left(\frac{\pi}{2}\right)}, e^{j(0)}, -e^{j\left(\frac{\pi}{2}\right)}, 1\right\} \qquad (2.2)$$
$$= e^{j\varphi_1}\left\{j, 1, j, -1, j, 1, -j, 1\right\}.$$

The chipping sequence of (2.2) is complex and multiplies the DQPSK symbol defined by φ_1. This is equivalent to traditional DSSS where the data symbol is multiplied by a real pseudorandom chipping sequence. The modulated waveform is complex, requiring the real and imaginary parts to be transmitted separately.

To obtain the 11-Mbps data rate, six bits (instead of the two bits used for the 5.5-Mbps data rate) are used to encode the phase angles φ_2, φ_3, and φ_4 as shown in Table 2, where the bits are grouped into pairs (d_2 and d_3, d_4 and d_5, d_6 and d_7). In both the 5.5-Mbps and 11-Mbps data rate formats, the first two data bits are DQPSK modulated (formulating the phase change φ_1).

Table 2 11 Mbps CCK Encoding [From Reference 9.]

Bits d_i and d_{i+1}	**Phase**
00	0
01	π/2
10	π
11	3π/2

The first pair of bits (d_0 and d_1) are used to encode φ_1 in accordance with Table 1. The second pair of bits (d_2 and d_3) are used to encode φ_2 in accordance with Table 2. The third pair of bits (d_4 and d_5) are used to encode φ_3 in accordance with Table 2, and the fourth pair of bits (d_6 and d_7) are used to encode φ_4 in accordance with Table 2. The resulting symbol is formed the same as in the 5.5-Mbps data rate case.

Since the CCK sequences are uncorrelated by definition, this forms a unique symbol for each group of four or eight data bits. The chip rate is kept at 11 Mcps, the same as the Standard 802.11-1997. The result is a signal that is contained within the same bandwidth but is able to transmit at a higher data rate. [9]

Another way to achieve the higher data rates required by the Standard 802.11b was accomplished by using packet binary convolutional coding (PBCC). Essentially, higher data rates are achieved by using convolutional coding combined with BPSK or QPSK modulation. The data is first encoded with a rate one-half convolutional encoder. The encoder output is then modulated using BPSK for the 5.5-Mbps data rate and QPSK for the 11-Mbps data rate. A pseudorandom sequence generator determines the BPSK or QPSK symbol constellation randomly by choosing one of two possible mapping schemes determined by one bit of the sequence generator. This encoding allows for a higher data rate while still achieving the required packet error rate. [9]

Also in September of 1999 the IEEE Standards Committee approved a high-speed standard for use in the 5 GHz unlicensed national information infrastructure (U-NII) bands using orthogonal frequency-division multiplexing (OFDM), Standard 802.11a [10]. This standard supports data rates of up to 54 Mbps. This standard later became the basis for the further extension of the 802.11 standard in the 2.4 GHz frequency range via the Standard 802.11g. Since the preamble and header formats of this standard are different than those used before, this format is not backwards compatible with the earlier IEEE Standard 802.11-1997 and IEEE Standard 802.11b.

The latest extension of the wireless local area network (LAN) Physical Layer standard (IEEE Standard 802.11) in the 2.4 GHz (Industrial, Scientific, and Medical) band was approved on 12 June 2003 as IEEE Standard 802.11g [11]. The purpose of this standard is to extend the original 802.11 standard to data rates greater than 20 Mbps in the 2.4 GHz band. This was achieved by using the OFDM technique used in IEEE Standard 802.11a to obtain data rates of up to 54 Mbps. PBCC used in Standard 802.11b was also used to obtain data rates of greater than 20 Mbps, reaching 22 Mbps and 33 Mbps. In addition, DSSS was combined with OFDM, which is referred to as DSSS-OFDM in the Standard 802.11g, to achieve 54-Mbps data rates without additional coordination from

those systems that are capable of using the Standard 802.11-1997 and Standard 802.11b data formats (DSSS-OFDM is also referred to as CCK-OFDM in some of the literature). This format uses Standard 802.11a OFDM to transmit the data portion of the packets, but uses the CCK-DSSS of Standard 802.11b to transmit the physical layer preamble and header information.

Since the preamble and the headers of a Standard 802.11g formatted packet using the extended rate are sent using OFDM signaling, the use of the extended rate Standard 802.11g is not backwards compatible with the Standard 802.11 and Standard 802.11b header and preamble formats. However, Standard 802.11g certified equipment must be able to support the other data formats of Standard 802.11b, enabling Standard 802.11g devices to work in the same networks as the Standard 802.11b devices. [11]

Although not generally cited in the group of wireless standards, IEEE Standard 802 [12] contains some of the basic requirements that govern not only the wireless standards but also all the local area network (LAN) and metropolitan area network (MAN) standards. This includes the error performance of a wireless network interface with other networks. As specified in the standard, the probability that a medium access control (MAC) Service Data Unit (MSDU) is not delivered correctly at a MAC service access point (MSAP) due to the operation of the Physical layer and the MAC protocol shall be less than 8×10^{-8} per octet of MSDU length. To find the allowed data bit error rate, note that the probability that eight consecutive bits are received without error is given by

$$\Pr(no\ error) = (1-p)^8 = 1-8\times10^{-8} \tag{2.3}$$

where p is the probability of an error of a single bit, or the desired bit error rate. Solving (2.3) for the bit error rate yields $p = 1\times10^{-8}$ as the allowed data bit error rate in wireless 802 compliant systems. However, as discussed in the following sections, the 802.11 standard required channel error rates for the physical layers are not as good as this specification. This requires the MAC layer to compensate for the decrease in error performance. This compensation is via the use of cyclic redundancy code error detection in the MAC layer header.

B. COMPATIBILITY BETWEEN 802.11 STANDARDS

With the addition of an OFDM modulation format to the 802.11 standard, a mechanism for preventing interference between stations operating in a service set at the same frequency that are not Standard 802.11g capable is required. While a Standard 802.11g extended rate physical layer (ERP) compliant system must be able to operate in an ad-hoc network with a possible combination of OFDM and DSSS systems, an interference problem arises when working with Standard 802.11-1997 or Standard 802.11b systems and Standard 802.11g systems operating using OFDM packets. The interference problem is a result of any non-OFDM station not being able to recognize the transmission of an OFDM signal that Standard 802.11g stations may be using to communicate with each other. Since the header of a Standard 802.11g OFDM transmission is an OFDM signal, non-Standard 802.11g compatible receivers will not recognize another station transmitting and may try and send a packet of their own after they sense that the channel is clear, resulting in a transmission collision between non-OFDM packets and OFDM packets. [11]

The 802.11 architecture mitigates collisions between transmitting stations using an algorithm called carrier sense multiple access with collision avoidance (CSMA/CA). For a station to transmit, it must sense the medium to determine if another station is transmitting. If the medium is not determined to be busy, both physically and virtually, then the transmission may proceed. A virtually busy channel is one in which a future transmission has been predicted, but not yet occurred. This allows replies to be sent by receiving stations without interference. A minimum interval is specified between packets that other stations must wait before attempting to transmit to allow for the controlling station (referred to as the point controller) to send required, contention-free traffic and also allow time for stations to send an immediate acknowledgement of received traffic. In addition, each station selects a random back-off interval to wait after the minimum wait period has been completed. This prevents all stations from attempting to transmit at the end of the mandatory wait period, the most likely time when transmission collision would otherwise occur. The 802.11 architecture incorporates modifications of this procedure by allowing stations in a service set to exchange short control frames, specifically request-to-send (RTS) and clear-to-send (CTS), prior to data transmission. [8]

There are three different frame types allowed in the 802.11 architecture. Management frames are used to support services of the system, such as association, authentication, and privacy. Control frames are used to support the delivery of data and management messages such as specification of the protocol used, the type of frame transmitted, duration of transmission, number of data fragments in the total transmission, and power management data frames are used to transmit the actual data. To allow for proper coordination among the different stations in a service set, all stations must be able to read the CTS and RTS control frames; therefore, these frames are sent in a format that is recognizable by all stations. This coordination is set up when stations first request to join a service set. This is accomplished through the association procedure where the station is associated with an access point (AP) and the authentication procedure where the station identifies itself to the AP. To join a service set, a station must first authenticate itself by exchanging authentication messages (management frame type) with the access point, which establishes the station's identity. Next the station associates itself with the AP, which gives the station's address to the AP and assigns an association ID to the station. It is in this association request and response that the AP specifies to the station what rates are supported in the basic service set and the station specifies to the AP what rates and capabilities the station supports. The association response is tailored to the format of the association request modulation format. That is, information on the service set parameters is tailored to the modulation type of the association request. [8]

The virtual carrier sense mechanism is achieved by distributing medium reservation information announcing the impending use of the medium. The exchange of RTS and CTS frames prior to the actual data frame is one means of distribution of this medium reservation information. The RTS and CTS frames contain a duration/ID field that defines the period of time that the medium is to be reserved to transmit the actual data frame and the returning acknowledge (ACK) frame. All stations within the reception range of either the originating station (which transmits the RTS) or the destination station (which transmits the CTS) will know of the medium reservation. The virtual carrier sense mechanism provided by the medium access control (MAC) layer is referred to as the Network Allocation Vector (NAV). The network allocation vector (NAV) maintains a prediction of future traffic on the medium based on duration information that is an-

nounced in RTS/CTS frames prior to the actual exchange of data and is used as an indicator for each station of time periods when the station will not transmit in order to avoid collisions with other stations. When a station first joins a service set, its network allocation vector is set to the maximum contention-free period value as specified by the point controller in a beacon message that is sent to all stations on a regular basis for housekeeping purposes. The network allocation vector is then modified as RTS and CTS and beacons are received. [8]

The result of these coordination issues is that, for Standard 802.11g capable stations operating in a mixed capability service set, only the data frame may be transmitted using OFDM. In addition, housekeeping to inform all other stations of the impending transmission must precede each data frame. [11]

C. IEEE STANDARD 802.11G TRANSMITTER

A Standard 802.11g compliant extended rate physical layer implements all the mandatory modes of the Standard 802.11a, except it uses the 2.4 GHz frequency band described in the Standard 802.11b. In addition, it is mandatory that all Standard 802.11g compliant equipment be capable of sending and receiving the short preamble as described in the Standard 802.11b. [11]

Three other changes implemented with the new Standard 802.11g apply to the DSSS transmission. This includes the ability to detect all the synchronization symbols of the new standard (including the ODFM symbols). Second, the error rate performance has been upgraded to achieve a frame error rate (FER) of less than 8×10^{-2} for a PSDU length of 1024 bytes at a received power level of -20 dBm measured at the antenna vice -10 dBm as specified in the Standard 802.11b [11]. Using the same method as before to calculate the required data bit error rate, we have

$$1-\mathrm{FER}=(1-p)^{8(1024)}. \tag{2.4}$$

Rearranging (2.4) and solving for p, the data bit error rate, we get

$$\begin{aligned}(1-p)^{8192}&=1-\mathrm{FER}\\ p&=1-\sqrt[8192]{(1-\mathrm{FER})}.\end{aligned} \tag{2.5}$$

Substituting $\text{FER} = 0.08$ into (2.5), the data bit error rate for DSSS-OFDM is then calculated to be

$$p = 1.0178 \times 10^{-5}. \tag{2.6}$$

Equation (2.6) gives the required error rate of the transmitted data. Finally, synchronization of the transmit oscillator and symbol clocks has also been upgraded so that the transmit frequency oscillator and the symbol clock oscillator must use the same reference oscillator.

Five other changes implemented with the new Standard 802.11g apply to the OFDM transmission. This includes the ability to detect all the synchronization symbols of the new standard (including DSSS). The transmit frequency accuracy and the symbol clock frequency tolerance has been changed from ± 20 ppm in the Standard 802.11a to ± 25 ppm. The error rate performance has been changed from a maximum packet error rate (PER) of 10% at a PSDU length of 1000 bytes at a receiver power level of -30dBm to a receiver power level of -20dBm [11]. Converting this to a required channel bit error rate, using the same method as above, we get

$$1 - \text{PER} = (1-p)^{8(1000)}. \tag{2.7}$$

Rearranging (2.7) and solving for p, we get

$$\begin{aligned} (1-p)^{8000} &= 1 - \text{PER} \\ p &= 1 - \sqrt[8000]{(1-\text{PER})}. \end{aligned} \tag{2.8}$$

Using the value of $\text{PER} = 0.1$ in (2.8) we obtain the required channel bit error rate for OFDM to be

$$p = 1.3170 \times 10^{-5}. \tag{2.9}$$

Equation (2.9) gives the error rate of the transmitted packet. The slot time, defined as the sum of the receiver-to-transmitter turnaround time, the MAC processing delay and the clear channel assessment detect time, which is the smallest wait time required during the back-off-window required to avoid collisions, is set to $20\ \mu\text{s}$, the same as required for the DSSS transmission. However, an option exists that allows this value to be reduced to the

9-μs value used by the Standard 802.11a if all stations in the service set support the Standard 802.11g. The short interframe spacing interval is set to 10 μs , the same used in the Standard 802.11b, but shorter than the 16-μs short interframe spacing interval required for the Standard 802.11a. [11]

The general block diagram of the transmitter for the OFDM physical layer is shown in Figure 2.

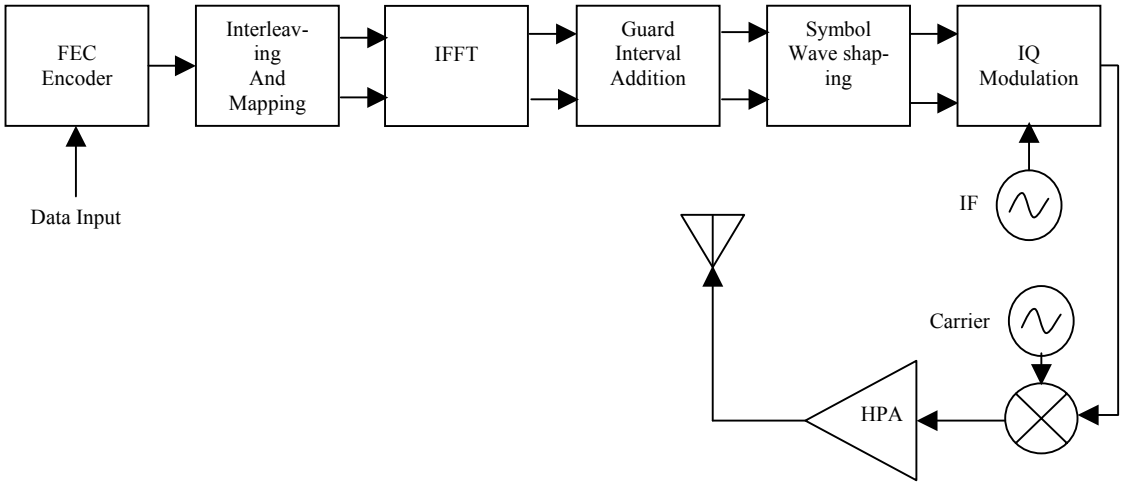

Figure 2 OFDM Transmitter [After Reference 10.]

The input data arrives at the physical layer from the MAC layer with header information and a frame check sequence (FCS) that consists of a 32-bit cyclic redundancy code (CRC) appended after the data to detect packet errors. This packet is called the MAC protocol data unit (MPDU) while it resides in the MAC layer, and the physical layer convergence protocol (PLCP) service data unit (PSDU) as it enters the physical layer. The frame body can contain a maximum of 2312 octets, or bytes, of data. With a header size of 30 bytes and a 4-byte CRC, the total maximum number of bytes arriving from the medium access control layer is 2346 bytes. This is a limitation placed on the size of the MPDU to support the requirements of the infrared transmission of the packet.

The actual limitation placed on OFDM transmission of the packet is 4095 bytes. If all the receivers in the service set support this MPDU size, then sending the larger packets can reduce the overhead. [8,10]

Once inside the physical layer a preamble, header, tail bits, and pad bits are added to the PSDU, and the entire packet is referred to as the PLCP protocol data unit (PPDU). The PPDU is the frame that is then transmitted over the wireless medium to the receiving station. The PPDU frame format for OFDM transmission is shown in Figure 3. [8]

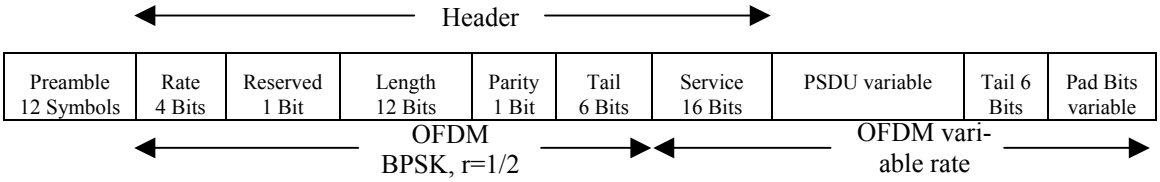

Figure 3 PPDU Frame Format [After Reference 10.]

The preamble for the OFDM signal consists of ten short training symbols consisting of one 8-μs long OFDM symbol, and two long training symbols consisting of two 4-μs OFDM symbols (including the guard interval) to aid the receiver in detecting and synchronizing the OFDM signal. The short training symbols are intended for initial signal detection, automatic gain control (AGC), and diversity selection. Also, a coarse frequency offset estimation and bit timing synchronization can be obtained from these symbols. The long training symbols are intended for channel estimation and fine frequency offset estimation. Use of this data considerably shortens the required time to detect and synchronize the received packet. [10]

The header information contains fields that specify the transmission rate of the PSDU, its length, and some transmitter initialization bits. This information is used after reception to specify the receive bit rate of the rest of the PPDU. [10]

The data field consists of a service field that initializes the data scrambler, the PSDU received from the MAC layer, and six tail bits that are added to the end of the PSDU to reset the convolutional encoder to the zero state. Finally, enough pad bits are added to the end so that the number of bits in the data field (Service field + PSDU + tail +

pad) is an integer multiple of required coded bits per OFDM symbol (N_{CBPS}). This ensures that a total of 48 values are entered into the Inverse Fast Fourier Transform (IFFT) modulator as data. [10]

The header is transmitted at 6 Mbps using binary phase-shift keying (BPSK) modulation and encoded with a rate $r = 1/2$ code to give it the best chance of detection. The data field is transmitted using one of the specified rates shown in Table 3 for the Standard 802.11g. [10]

Table 3 OFDM Rate Dependent Parameters [From Reference 10.]

Data rate (Mbits/s)	**Modulation**	**Coding rate (*r*)**	**Coded bits per sub-carrier (N_{BPSC})**	**Coded bits per OFDM symbol (N_{CBPS})**	**Data bits per OFDM symbol (N_{DBPS})**
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

The short preamble and header format was first used in the Standard 802.11b as an option for networks that did not need to support the older standard with the long preamble and header. The format is intended to increase data throughput in applications where this is important, such as video and audio transmission. However, this format is only specified for the DSSS modulation format. The long PPDU is shown in Figure 4, and the short PPDU is shown in Figure 5. [9]

The reduction in size is from a total of 192 bits for the long format preamble and header to a total of 120 bits for the short format preamble and header, a reduction of 37.5%. Standard 802.11g is required to support this short PPDU format. [9]

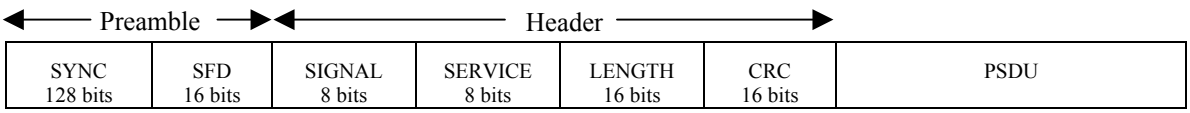

Figure 4 Long PPDU Format [After Reference 9.]

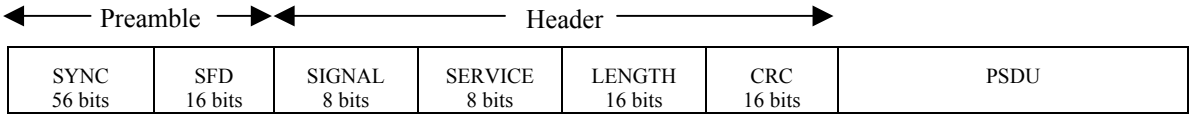

Figure 5 Short PPDU Format [After Reference 9.]

Once the service field is appended to the received data in the PSDU packet, it is scrambled and passed through a constraint length seven, rate $r = 1/2$ convolutional encoder. The required coding rates, if different than $r = 1/2$, can be obtained by puncturing the data to remove some of the coded bits. This removal of some of the coded bits while keeping the number of data bits constant increases the code rate (data bits divided by coded bits) to the desired value. [10]

After coding, the data is interleaved to randomize the errors in the coded data bits introduced by the wireless channel. This prevents consecutive coded data bits from being affected by noise bursts in the channel, which improves the performance of the decoding process. Standard 802.11g uses a two-stage process to interleave the data. The first stage ensures that adjacent coded bits are mapped onto nonadjacent sub-carriers. This is accomplished by reading the coded data bits into a standard ($N_{CBPS}/16$ by 16) block interleaver. The second stage ensures that adjacent coded bits are mapped alternately onto less and more significant bits of the constellation, avoiding long runs of low reliability LSB bits. The second stage is actually not performed for the first four data rates of Table 1, since the constellation size for BPSK is one bit and the constellation size for QPSK is two bits. In these cases, alternating the bits will not change the constellation mapping. However, for 16-QAM the first stage of interleaving leaves all data bits on the same row of the block interleaver mapped into the same bit of the 16-QAM constellations. If every

other column of the block interleaver has every other bit exchanged, this eliminates adjacent bits in the same row from being mapped to the same constellation bit. The interleaving process is described with the help of Figure 6. In this example, the 24-Mbps data rate is used. [10]

A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	K1	L1	M1	N1	O1	P1
A2	B2	C2	D2	E2	F2	G2	H2	I2	J2	K2	L2	M2	N2	O2	P2
A3	B3	C3	D3	E3	F3	G3	H3	I3	J3	K3	L3	M3	N3	O3	P3
A4	B4	C4	D4	E4	F4	G4	H4	I4	J4	K4	L4	M4	N4	O4	P4
A5	B5	C5	D5	E5	F5	G5	H5	I5	J5	K5	L5	M5	N5	O5	P5
A6	B6	C6	D6	E6	F6	G6	H6	I6	J6	K6	L6	M6	N6	O6	P6
A7	B7	C7	D7	E7	F7	G7	H7	I7	J7	K7	L7	M7	N7	O7	P7
A8	B8	C8	D8	E8	F8	G8	H8	I8	J8	K8	L8	M8	N8	O8	P8
A9	B9	C9	D9	E9	F9	G9	H9	I9	J9	K9	L9	M9	N9	O9	P9
A10	B10	C10	D10	E10	F10	G10	H10	I10	J10	K10	L10	M10	N10	O10	P10
A11	B11	C11	D11	E11	F11	G11	H11	I11	J11	K11	L11	M11	N11	O11	P11
A12	B12	C12	D12	E12	F12	G12	H12	I12	J12	K12	L12	M12	N12	O12	P12

Figure 6 Standard 802.11g Stage One Interleaver.

The rows are numbered one through twelve, and the columns are marked A – P. The data is read in sequentially across each row. Since $N_{CBPS} = 192,$ a total of twelve rows of sixteen columns are used to fill the block interleaver. The first stage of the interleaver reads out the data in columns. If the second stage of the interleaver were not used, the 16-QAM mapping would be as shown in Figure 7. [10]

Note that row one only shows up in bit b_0 of the 16-QAM mapping, row two only shows up in bit b_1 of the 16-QAM mapping, etc. This means entire rows are mapped to the same bit in the QAM mapping.

To randomize the distance between symbols, the second stage of the interleaver swaps coded bits between rows in every even column of the block interleaver. The result

is that only every other bit in each row is mapped to the same bit in the symbol constellation. Figure 8 shows the block interleaver after data bits have been swapped by the second stage of the interleaver. [10]

Symbol	b_0	b_1	b_2	b_3	Symbol	b_0	b_1	b_2	b_3	Symbol	b_0	b_1	b_2	b_3
1	A1	A2	A3	A4	17	F5	F6	F7	F8	33	K9	K10	K11	K12
2	A5	A6	A7	A8	18	F9	F10	F11	F12	34	L1	L2	L3	L4
3	A9	A10	A11	A12	19	G1	G2	G3	G4	35	L5	L6	L7	L8
4	B1	B2	B3	B4	20	G5	G6	G7	G8	36	L9	L10	L11	L12
5	B5	B6	B7	B8	21	G9	G10	G11	G12	37	M1	M2	M3	M4
6	B9	B10	B11	B12	22	H1	H2	H3	H4	38	M5	M6	M7	M8
7	C1	C2	C3	C4	23	H5	H6	H7	H8	39	M9	M10	M11	M12
8	C5	C6	C7	C8	24	H9	H10	H11	H12	40	N1	N2	N3	N4
9	C9	C10	C11	C12	25	I1	I2	I3	I4	41	N5	N6	N7	N8
10	D1	D2	D3	D4	26	I5	I6	I7	I8	42	N9	N10	N11	N12
11	D5	D6	D7	D8	27	I9	I19	I11	I12	43	O1	O2	O3	O4
12	D9	D10	D11	D12	28	J1	J2	J3	J4	44	O5	O6	O7	O8
13	E1	E2	E3	E4	29	J5	J6	J7	J8	45	O9	O10	O11	O12
14	E5	E6	E7	E8	30	J9	J10	J11	J12	46	P1	P2	P3	P4
15	E9	E10	E11	E12	31	K1	K2	K3	K4	47	P5	P6	P7	P8
16	F1	F2	F3	F4	32	K5	K6	K7	K8	48	P9	P10	P11	P12

Figure 7 Mapping of First Interleaver Stage.

The resulting 16-QAM mapping is shown in Figure 9. Note that row one is now mapped alternately into bit b_0 and b_1, avoiding long runs of bits mapped to the same bit.

The constellation mapping uses standard Gray-coded BPSK, QPSK, 16-QAM, and 64-QAM mapping, depending on the requested rate, as specified in Table 3. The complex output of the mapping is multiplied by a normalization factor to achieve the same average power for all mappings, which accounts for changes in modulation during the transmission of the packet. [10]

A1	B2	C1	D2	E1	F2	G1	H2	I1	J2	K1	L2	M1	N2	O1	P2
A2	B1	C2	D1	E2	F1	G2	H1	I2	J1	K2	L1	M2	N1	O2	P1
A3	B4	C3	D4	E3	F4	G3	H4	I3	J4	K3	L4	M3	N4	O3	P4
A4	B3	C4	D3	E4	F3	G4	H3	I4	J3	K4	L3	M4	N3	O4	P3
A5	B6	C5	D6	E5	F6	G5	H6	I5	J6	K5	L6	M5	N6	O5	P6
A6	B5	C6	D5	E6	F5	G6	H5	I6	J5	K6	L5	M6	N5	O6	P5
A7	B8	C7	D8	E7	F8	G7	H8	I7	J8	K7	L8	M7	N8	O7	P8
A8	B7	C8	D7	E8	F7	G8	H7	I8	J7	K8	L7	M8	N7	O8	P7
A9	B10	C9	D10	E9	F10	G9	H10	I9	J10	K9	L10	M9	N10	O9	P10
A10	B9	C10	D9	E10	F9	G10	H9	I10	J9	K10	L9	M10	N9	O10	P9
A11	B12	C11	D12	E11	F12	G11	H12	I11	J12	K11	L12	M11	N12	O11	P12
A12	B11	C12	D11	E12	F11	G12	H11	I12	J11	K12	L11	M12	N11	O12	P11

Figure 8 Standard 802.11g Stage Two Interleaver.

After mapping, the data symbols are partitioned into groups of 48, mixed with four pilot symbols, and then used as the inputs of an IFFT. The pilot symbols are scrambled prior to mixing to avoid the generation of spectral lines. The size of the IFFT is not specified, but a size of 64 is normally used, since this is the next power of two above 52 (48 data symbols and four pilot symbols). The output of the IFFT is cyclically extended over a guard interval to prevent intersymbol and intercarrier interference (ISI and ICI). This output is windowed to smooth the data transitions, heterodyned to the carrier frequency, amplified, and transmitted with a signal bandwidth of 16.6 MHz. [10]

The output of one OFDM symbol can be represented mathematically as [10]

$$r_{DATA,n}(t) =$$

$$w_{TSYM}(t)\left(\sum_{k=0}^{N_{SD}-1} d_{k,n}\exp\left(j2\pi M(k)\Delta_F\left(t-T_{GI}\right)\right)+p_{n+1}\sum_{k=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} P_k\exp\left(j2\pi k\Delta_F\left(t-T_{GI}\right)\right)\right) \quad (2.10)$$

Symbol	b_0	b_1	b_2	b_3	Symbol	b_0	b_1	b_2	b_3	Symbol	b_0	b_1	b_2	b_3
1	A1	A2	A3	A4	17	F6	F5	F8	F7	33	K9	K10	K11	K12
2	A5	A6	A7	A8	18	F10	F9	F12	F11	34	L2	L1	L4	L3
3	A9	A10	A11	A12	19	G1	G2	G3	G4	35	L6	L5	L8	L7
4	B2	B1	B4	B3	20	G5	G6	G7	G8	36	L10	L9	L12	L11
5	B6	B5	B8	B7	21	G9	G10	G11	G12	37	M1	M2	M3	M4
6	B10	B9	B12	B11	22	H2	H1	H4	H3	38	M5	M6	M7	M8
7	C1	C2	C3	C4	23	H6	H5	H8	H7	39	M9	M10	M11	M12
8	C5	C6	C7	C8	24	H10	H9	H12	H11	40	N2	N1	N4	N3
9	C9	C10	C11	C12	25	I1	I2	I3	I4	41	N6	N5	N8	N7
10	D2	D1	D4	D3	26	I5	I6	I7	I8	42	N10	N9	N12	N11
11	D6	D5	D8	D7	27	I9	I19	I11	I12	43	O1	O2	O3	O4
12	D10	D9	D12	D11	28	J2	J1	J4	J3	44	O5	O6	O7	O8
13	E1	E2	E3	E4	29	J6	J5	J8	J7	45	O9	O10	O11	O12
14	E5	E6	E7	E8	30	J10	J9	J12	J11	46	P2	P1	P4	P3
15	E9	E10	E11	E12	31	K1	K2	K3	K4	47	P6	P5	P8	P7
16	F2	F1	F4	F3	32	K5	K6	K7	K8	48	P10	P9	P12	P11

Figure 9 Mapping of Second Interleaver Stage.

where $n =$ the symbol number, $w_{TSYM} =$ the windowing function of duration time T equal to the symbol duration, $N_{SD} =$ the number of data sub-carriers (48), $d_{k,n} =$ the complex number corresponding to sub-carrier k of OFDM symbol n, $\Delta_F =$ the sub-carrier frequency spacing (0.3125 MHz), $T_{GI} =$ the guard interval duration (0.8 μs), $p_n =$ the scrambling sequence, $P_k =$ the pilot sub-carrier sequence, $N_{ST} =$ the total number of sub-carriers (52), and

$$M(k) = \begin{cases} k-26 & 0 \le k \le 4 \\ k-25 & 5 \le k \le 17 \\ k-24 & 18 \le k \le 23 \\ k-23 & 24 \le k \le 29 \\ k-22 & 30 \le k \le 42 \\ k-21 & 43 \le k \le 47 \end{cases}. \tag{2.11}$$

Equation (2.11) defines the mapping from the logical sub-carrier number 0 to 47 into frequency offset index -26 to 26, while skipping the pilot sub-carrier locations and the 0th (dc) sub-carrier.

The transmitted packet is just a concatenation of the data symbols from (2.10) (neglecting the preamble and the header) and can be written mathematically as [10]

$$r_{DATA}(t) = \sum_{n=0}^{N_{SYM}-1} r_{Data,n}\left(t - nT_{SYM}\right) \tag{2.12}$$

where N_{SYM} = the total number of OFDM symbols transmitted, and T_{SYM} = the symbol interval (4 μs). Equation (2.12) represents the signal from the OFDM transmitter.

D. IEEE STANDARD 802.11G OFDM RECEIVER

An extended rate physical layer has the capability to detect both the extended rate preambles and the preambles from a Standard 802.11b or Standard 802.11 compliant system whenever a clear channel assessment is requested. Since the protection mechanism is not required in all cases, the extended rate physical layer must be able to detect all preamble types at all times. [11]

The general block diagram of the receiver for the OFDM physical layer is shown in Figure 10. For the most part, the receiver blocks are simply inverses of their counterparts in the transmitter. However, synchronization plays a significant role in the detection of the incoming packet. In addition, careful application of the inverse process is required in many of the receiver blocks to ensure a reproduction of the original signal.

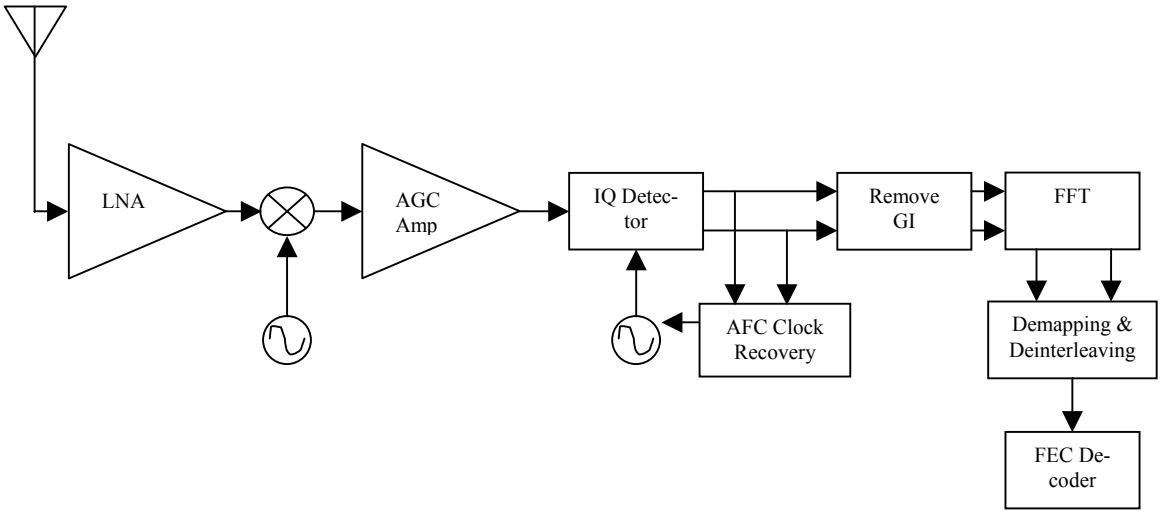

Figure 10 OFDM Receiver [From Reference 10.]

Upon packet detection and system synchronization, the data in the guard interval is removed and then the remaining data is demodulated using the FFT. The received data can be immediately de-mapped if hard decision decoding is used, or the FFT output can be used in a soft decision-decoding scheme to improve system error rate performance. For an excellent discussion of the effect of error correction coding, including the effects of soft decision decoding, see the discussion in [5].

De-interleaving is simply the inverse of the interleaving process, however this is one block that requires the incoming data stream to be synchronized in the original block of coded bits per OFDM symbol (N_{CBPS}). This means the receiver must know exactly when the data stream starts, or the data will simply be jumbled by the de-interleaver.

Prior to the decoder, the data must be de-punctured in those cases where the code rate was increased to accommodate increased data rates. Inserting the bits that were erased during the puncturing process also requires synchronization with the incoming data stream. Inserting the erasure bits into the wrong positions will significantly affect the performance of the decoder circuit.

The convolutional decoding circuit of the receiver can use one of three possible methods [13]: sequential algorithms, threshold decoding algorithms, or the Viterbi decoding algorithm. Sequential algorithms provide fast, but suboptimal, decoding for convolu-

tional codes, and they are mostly used in applications where the coding constraint length is greater than ten. Threshold decoding algorithms use a series of parity check equations in a voting scheme, but are also suboptimal for convolutional codes. The Viterbi algorithm is a maximum-likelihood and a maximum a posteriori decoding algorithm. Since the constraint length of the convolutional encoder used in the transmitter is seven, this is the algorithm that is used in most Standard 802.11a and Standard 802.11g receivers. A significant consideration when using the Viterbi decoding algorithm is the path length used before a decoding decision is made. The longer the path length, the more likely the decoding solution will merge with the correct result, but the amount of memory required to store the path taken and the path metrics becomes too large to be practical. In practice, the maximum-likelihood path can be determined in a relatively small path length ($\leq 2^{\text{contraint length}}$). [13]

E. OPTIONAL STANDARD 802.11G MODES OF OPERATION

Two of the original proposals for the Standard 802.11g that did not get adopted as the primary mode in the standard but were later adopted as optional modes of operation for the new Standard 802.11g were a design from Texas Instruments called Packet Binary Convolutional Code (PBCC) and a design from Intersil Corporation called direct sequence spread spectrum orthogonal frequency-division multiplexing (DSSS-OFDM) [14]. PBCC does not use orthogonal frequency-division multiplexing, and DSSS-OFDM uses partial OFDM.

The basic block diagram of the PBCC modulation scheme is shown in Figure 11.

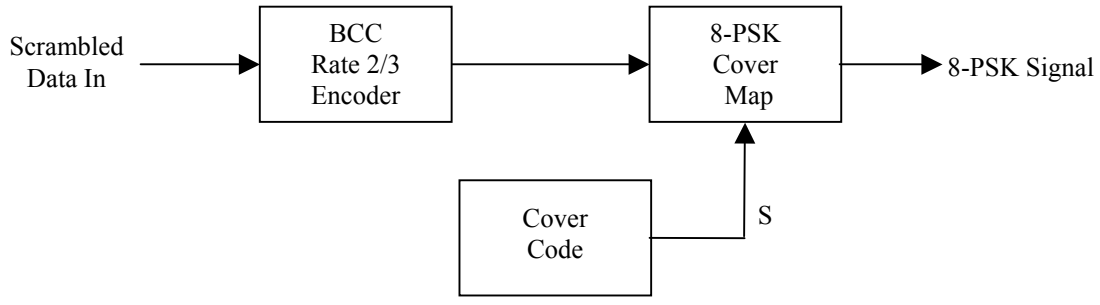

Figure 11 PBCC Bock Diagram [From Reference 9.]

Texas Instruments reports a 3-dB coding gain for this configuration over the competing CCK configuration. The data is passed through a 2^8 state (eight memory element) binary convolutional encoder (BCC) with a coding rate of $r = 2/3$. This is a change from the encoder used in the Standard 802.11b, which used a 2^6 state (six memory element) encoder with a coding rate of $r = 1/2$. The output of the convolutional encoder is mapped onto an 8-PSK constellation using three of the coded bits. In Standard 802.11b, the encoder output is mapped onto a BPSK constellation for the 5.5-Mbps data rate and a QPSK constellation for the 11-Mbps data rate. The mapping, however, is not constant. The mapping is determined pseudo randomly by the cover code, a 256-bit sequence. This means that there are two possible ways to map the coded data, depending on the value of the cover code bit, S. Figure 12 shows the two possible mappings for the 8-PSK modes. [11]

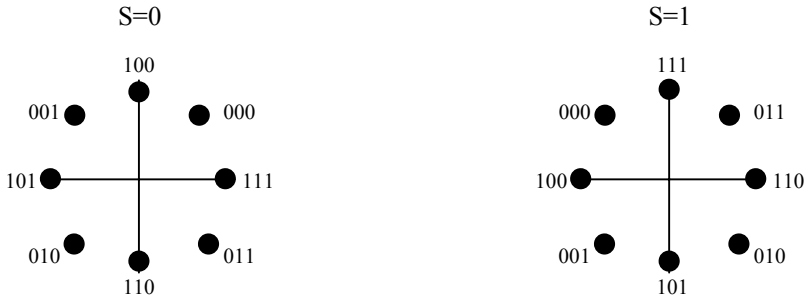

Figure 12 PBCC Constellation Mapping [From Reference 11.]

This mapping is used for both the 22-Mbps and the 33-Mbps data rates offered by the PBCC mode. The increased data rate is accomplished by changing the clock that drives the PBCC circuit from 11 Msymbols/s to 16.6 Msymbols/s. Since the preamble is sent using an 11-Mbps signal to make it backwards compatible with the Standard 802.11b standard, the 33-Mbps rate inserts a clock switching sequence between the preamble and the PSDU. This allows the clock to reset and resynchronize before receiving the data. [11]

The PBCC coding scheme is very similar to using Trellis Coded Modulation (TCM) with interleaving. TCM also uses convolutional encoding and multidimensional

signal constellations to modulate a signal. Figure 13 shows a typical TCM coding scheme. This coding scheme matches the PBCC scheme of Figure 11 if all data inputs are passed through the convolutional encoder (i.e., d_{k+1} through d_m are all set to zero in Figure 13). The cover sequence in PBCC acts as a random interleaver, preventing long strings of similar data bits from being mapped to the same constellation symbol. [13]

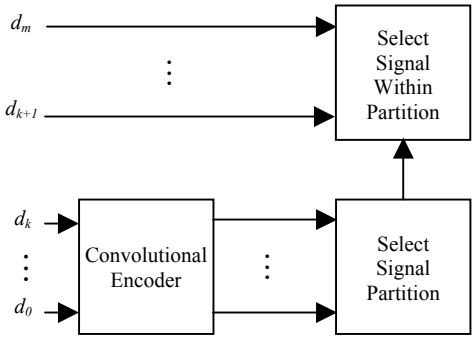

Figure 13 Trellis Coded Modulation Block Diagram [After Reference 13.]

The intent of using DSSS-OFDM was to extend the data rates to those offered by OFDM while reusing the preambles associated with the Standard 802.11 and Standard 802.11b. Since the preamble associated with the new data rate is the same preamble used with the old data rates, no protection mechanism to prevent transmission collisions is required and all data rates will understand the preamble without modification. The DSSS-OFDM option uses the complementary code-keying mode of the Standard 802.11b with the physical layer service data unit (PSDU) packaged as a modified OFDM packet. Figure 14 shows the PSDU for the DSSS-OFDM option. [11]

Long Training Sequence	OFDM Signal Field	OFDM Data Symbols	Signal Extension

Figure 14 DSSS-OFDM PSDU [From Reference 11.]

The long training sequence, the signal field and the data symbols are the same as those described in Figure 3.The signal extension is used to provide additional processing time (specifically the convolutional decoding) for the OFDM demodulator and is a period of no transmission. [11]

The difficult part of DSSS-OFDM is the transition from a single-carrier DSSS system to a multiple carrier OFDM system without forcing any parameter reacquisition. The parameters of concern are spectrum, power, timing, frequency and phase. [11]

Pulse shaping and windowing are used to achieve spectrum coherency. DSSS-OFDM uses a *sinc* pulse and a Hanning window to approximate the spectrum of the OFDM signal and provide frequency coherency within the packet. Timing coherency is achieved by noting that the 11-MHz clock of DSSS and the 20-MHz clock of OFDM align at every $1\ \mu s$ boundary. This means that both sequences line up after every eleven DSSS data symbols and 20 OFDM symbols. Simply replacing the DSSS symbol with the OFDM symbol at this $1\ \mu s$ boundary achieves timing alignment. Power coherency is achieved by requiring the transmit power levels for both signals to be the same. The carrier frequency is kept coherent by using the same carrier waveform for both signal types. To achieve phase coherency, the phase of the first OFDM symbol must be 45 degrees more than the phase of the last DSSS Barker symbol. Since all Barker symbols are transmitted at a phase of ± 45 degrees or ± 135 degrees to maximize transmitted power, the first OFDM symbol must be multiplied by ± 1 or $\pm j$, respectively, to achieve the required phase. [11]

The extension of the 802.11 standard to data rates above 20 MHz involves several aspects of previous 802.11 standards. Standard 802.11g brings all of these aspects together to achieve the required data rates and still maintain backwards compatibility with previous standards.

With the details of the Standard 802.11g specified, the next chapter will show how the OFDM portion of the Standard 802.11g can be simulated in order to study optimum receiver and transmitter designs.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SYSTEM VIEW SIMULATION

A. SYSTEM VIEW SYSTEM ANALYSIS TOOL

The software chosen for the synchronization simulation was SystemView, developed by Elanix Incorporated. SystemView is a systems analysis simulator that provides a graphical user interface that can potentially save significant amounts of time required to build and run engineering and scientific simulations. No code writing or command entry is required by the user to run a system simulation. An entire system can be conceived, designed and tested using only the mouse. This simulation software is designed for use with the Microsoft Windows family of operating systems.

The basic system building blocks are called tokens. Each token performs a specific function. SystemView allows general parameter entries for each token, allowing for significant flexibility in system design. In addition to the parameters allowed for each individual token, a large number of system timing parameters can be specified, allowing for various simulation run scenarios depending on the results of the outputs of specified tokens. In addition, the system includes filter design in the operator token's linear filter area.

SystemView uses two windows. The system window is where the system is designed and the various tokens can be accessed. In addition to the rich library of tokens provided by SystemView, custom tokens can be developed using C++ or C programming. The designer can simply develop a function that executes once during each cycle of the system clock and use that function as a token in the project design. SystemView also allows what it calls meta-tokens which allow the consolidation of several tokens into a single token. This simplifies the look of complex circuits and allows for modular design of complex systems. The second window is the analysis window where all the graphs and statistics of the data within the system can be analyzed. Analysis can be performed in the time and frequency domains. An analysis calculator is included which allows various calculations (e.g., correlation) to be performed on the results of the data runs.

Despite its potential for allowing quick and easy system design and analysis, SystemView can be challenging to employ. The biggest problem is the limited documenta-

tion provided. Even seasoned engineers with significant background working with problems solved by the tokens will have a difficult time understanding exactly how the token arrives at its final value. The equations used are often left out of the token descriptions. In addition, the exact effect of the input parameters on the token output is often in question. Only after repeated trials are many of the token characteristics mastered. This tends to defeat the advantages that System View could have over a programming simulation, such as MatLab.

A second challenging aspect of using SystemView is tracking the clock for each token. The maximum clock speed is specified before running the system simulation, but this is only the maximum speed of any part of the system, not the clock that runs the entire system. For example, if the number of output bits is greater than the number of input bits for a token, the system clock will speed up at that token output to get the data out in the same amount of time that it takes to input the data. This is a particular challenge with the multiplexer and de-multiplexer tokens, which place additional constraints on the clock speed of each of their inputs. Each token, in general, reads one data point at each clock cycle. However, source tokens output data at the system (maximum) clock frequency and, therefore, need to be sampled in most cases to allow the proper input to follow-on tokens.

A third challenging aspect of using SystemView is the delay introduced by some tokens in the data flow. Some tokens will not output any data until a certain number of data points have been entered. A good example of this is the de-multiplexer token. The output is zero until all of the inputs have read in their data points. Once this is completed, the output is enabled. System View also inserts a one-sample delay at the output of all feedback loops. While some delay is natural in circuit design, the fact that not all tokens have a delay can make design somewhat challenging. In addition, adding a token with built-in delay in a circuit where the system timing is critical can have a significant effect on the operation of the system.

After SystemView has been mastered, design is quick and easy. This allows a detailed analysis of complicated systems in a relatively short amount of time, and it is the basis for the OFDM analysis used in this thesis. The final design, once developed and

analyzed can also be used to develop a Field Programmable Gate Array (FPGA) implementation. FPGA implementation holds the promise of rapid design, analysis and implementation of optimum designs.

The system designed in this thesis is used to support the analysis of the different OFDM synchronization techniques, and it is based on a design developed by Elanix in its web site list of potential applications of System View as Application Note 140 [15]. The entire list of application notes can be view on the Elanix web site at http://www.elanix.com/html/products.asp.

B. TRANSMITTER

The entire SystemView transmitter circuit is shown in Figure 15. The transmitter simulation uses a data stream of a continuous random bit sequence to allow accurate bit error rate measurements on the order of 1×10^{-5}. The number of data samples required to generate bit error rate curves is at least ten times the inverse of the anticipated bit error rate. This means that an anticipated bit error rate of 1×10^{-5} requires a minimum of $10\times\left(1\times10^{5}\right)$ or 1×10^{6} data samples. A minimum of two system samples per clock period are required to generate an alternating clock signal; otherwise the clock is always sampled at the same phase and its value never changes. This transmitter circuit uses the 36-Mbps data rate that requires a maximum system clock frequency of 72 MHz. To detect an error rate of 1×10^{-5}, 2×10^{6} system samples are used.

If the circuit is modified to send the OFDM packet on a carrier frequency of 2.5 GHz, the maximum system clock rate will be 5 GHz. Since about one hundred forty $\left(2\times2.5\text{ GHz}/36\text{ MHz}\right)$ system clock cycles are now needed to generate one data sample, the number of required system samples jumps to 140×10^{6} (an increase by a factor of 35). Since it takes roughly 4.5 minutes to run 1,000,000 (2^{20}) system sample points, it will take approximately $4.5\times140=630$ minutes on an Intel Pentium IV processor running at 2.0 GHz, or 10 hours and 30 minutes to run the simulation with the signal modulated at the carrier frequency. This is too long to run a reasonable number of simulations, and it gives no additional insight. Therefore, all the simulations run in this Chapter of the thesis

were done at baseband. Only the baseband signal is used in the design to avoid lengthy run times with the higher frequencies required in the Standard 802.11g.

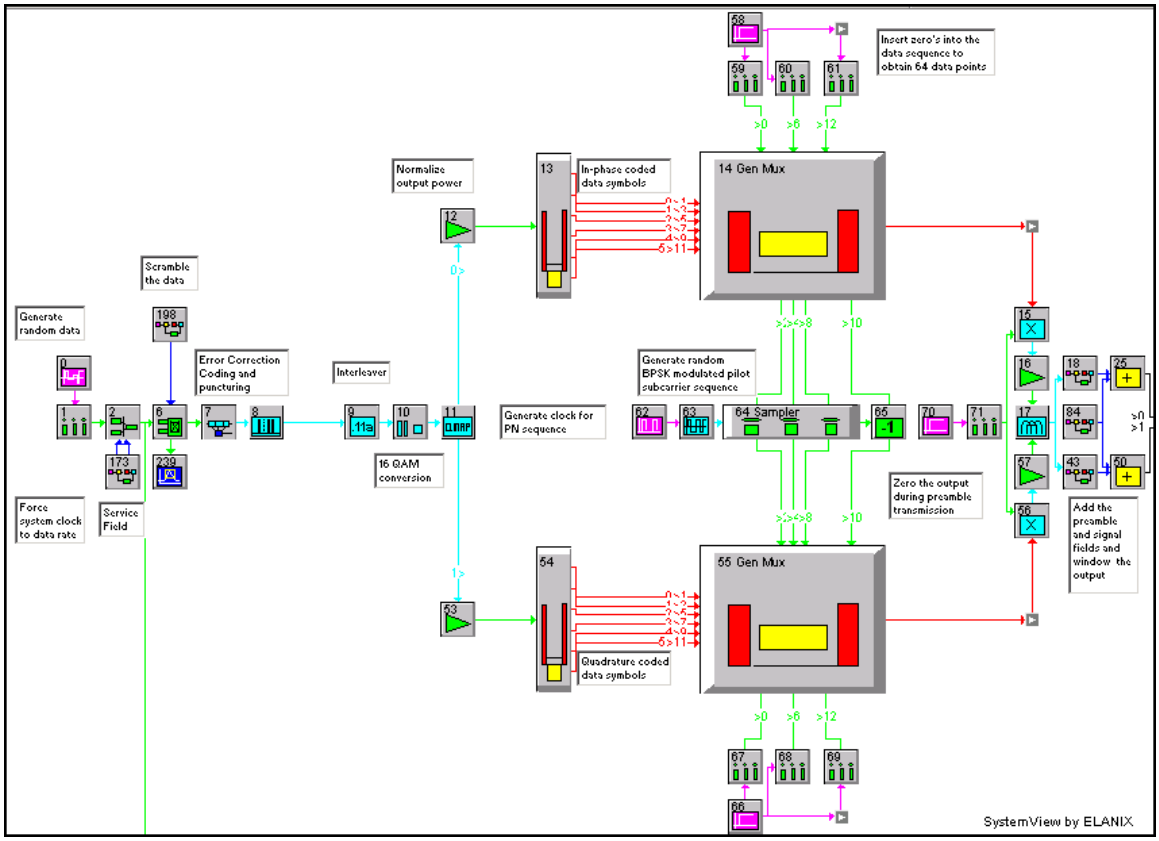

Figure 15 OFDM Transmitter Circuit.

1. Data Generation

The data generation portion of the transmitter circuit is shown in Figure 16. It includes the random data generator and the service field generator, which is the portion of the Standard 802.11g header that performs scrambler initialization in the receiver. This portion of the packet header is always transmitted at the data rate, requiring it to be inserted at the beginning of the data. The service field MetaSystem is expanded in Figure 17. It generates a short pulse (16 data bits long) at one output (token 180) to control the switch between the data generation token (token 0) and the service field generation token (token 173). The second output (token 181) generates the service field in accordance to Reference [10] consisting of sixteen zeros.

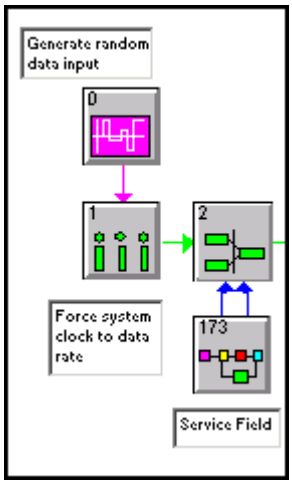

Figure 16 Transmitter Data Generation Circuit.

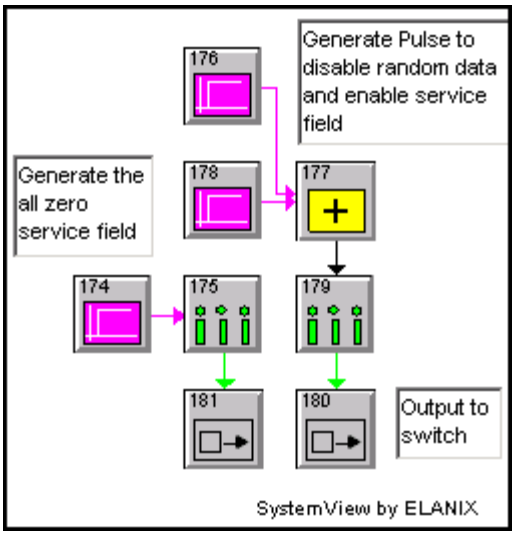

Figure 17 Service Field MetaSystem.

The parameters used to generate the pseudorandom sequence representing the packet data are shown in Figure 18. The data rate is set to 36 Mbps, which is the data rate used in both the example of Reference [10] and in the Application Note 140 from Elanix [15]. This data rate shows the most aspects of the OFDM portion of the Standard 802.11g as compared to the other possible data rates. The amplitude and offset are used to produce a square wave where bit zero is represented by zero volts and a bit one is represented by

one volt. Since this is a binary sequence, only two levels are used. Figure 19 shows the parameters for the sampler token (token 1).

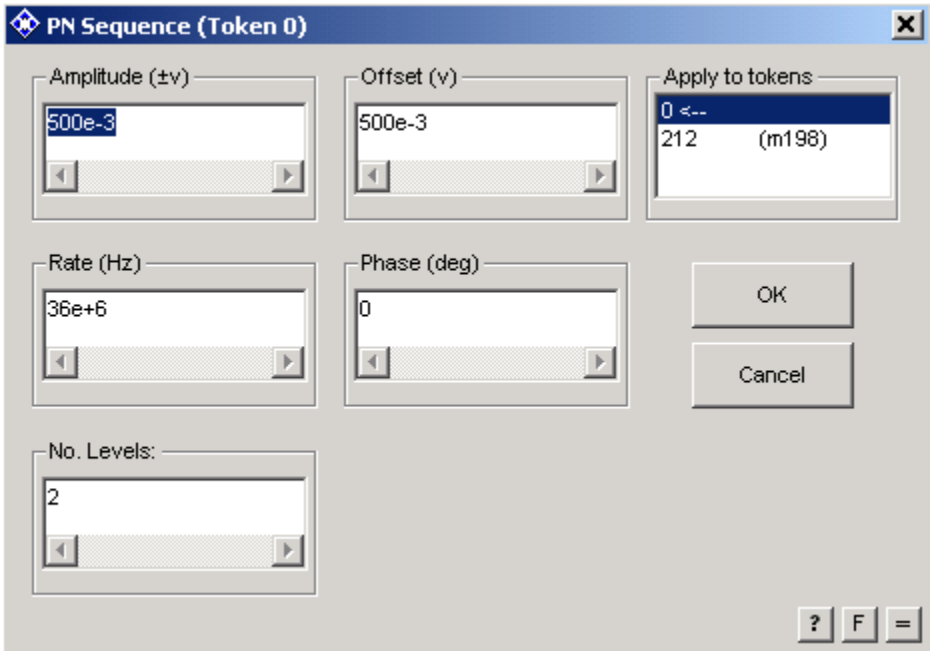

Figure 18 Pseudorandom Token Parameters.

The sampler is required to change the clock seen by the follow-on tokens to the data rate of 36 Mbps. Otherwise two data bits would be entered for every one data bit generated, since the system clock is running at 72 MHz. The same parameters are used for all the samplers positioned after a source token since all sources are generated at bit rates equal to the system time. This can also be seen after the unit step functions of the service field MetaSystem.

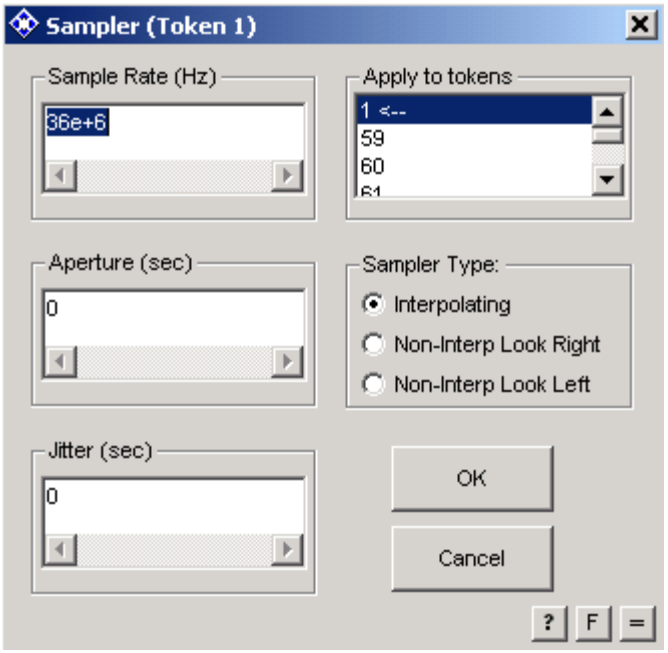

Figure 19 Sampler Token Parameters.

Figure 20 shows the parameters of one of the step function generators used in the service field MetaSystem. This step function and the step function of token 176 are combined to produce a pulse of duration equal to 16 times the bit duration (i.e., output = $u(0)-u(\tau)$). Figure 21 shows the parameters for the switch used to insert the service field. The switch can have up to twenty inputs; however, only two are used in this case. The control voltage selects which input is connected to the output. A control voltage of zero volts connects the service field to the output. A control voltage of one volt connects the random data to the output.

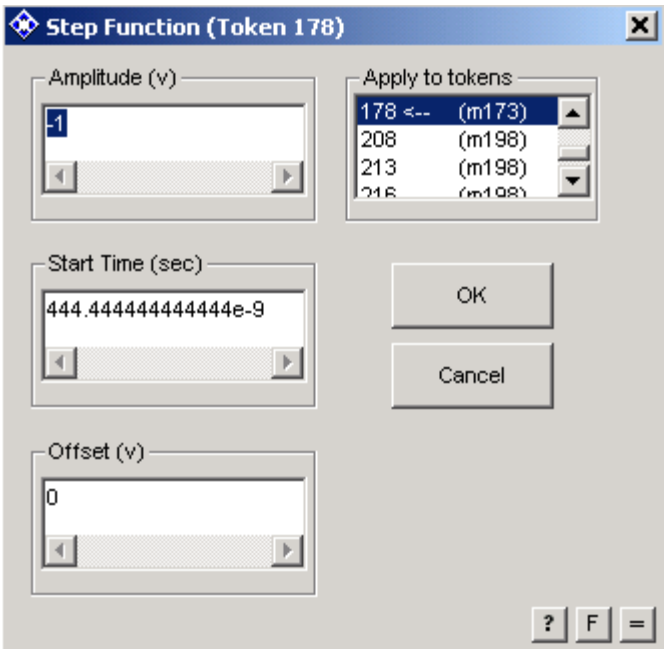

Figure 20 Step Function Parameters.

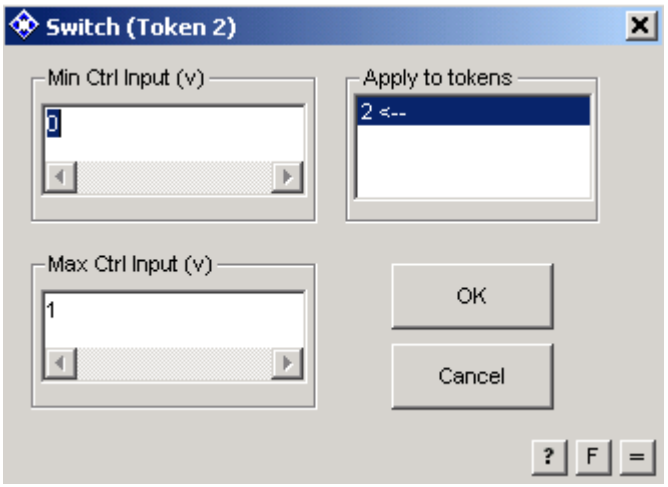

Figure 21 Switch Token Parameters.

Figure 22 shows an example of the data produced by the data generation circuit with each circle representing one data point. The depicted signal represents 16 service field bits followed by the random data bits. Note the sixteen zeros leading the random data.

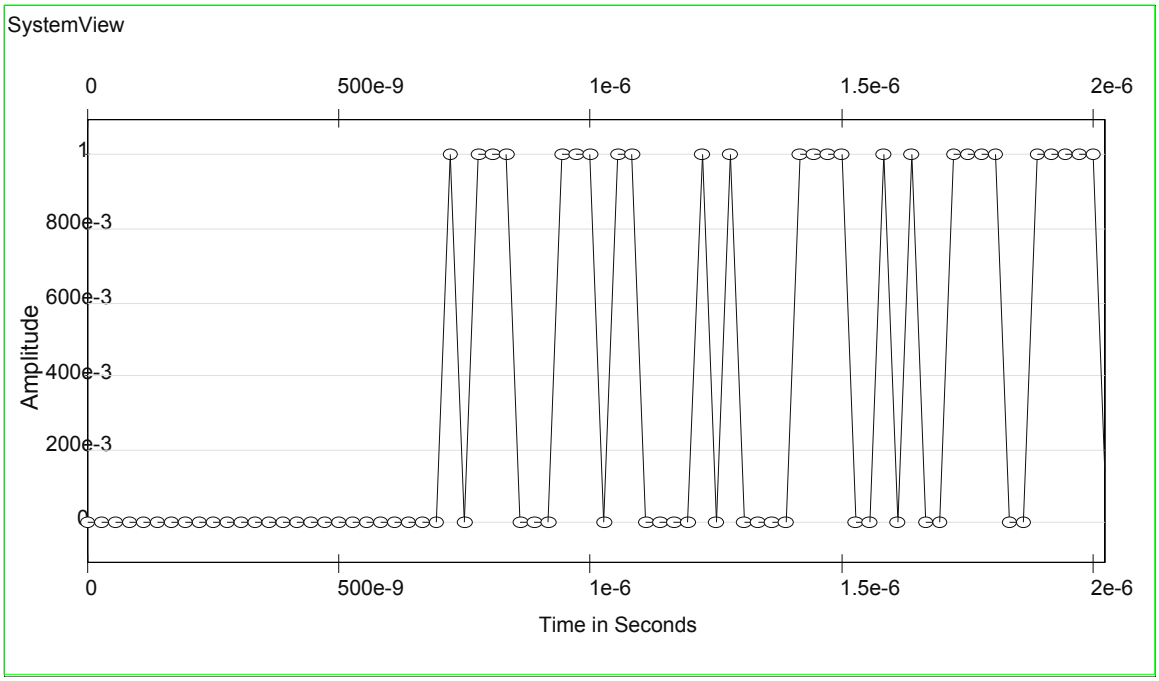

Figure 22 Example Run of the Data Generation Circuit.

2. Coding and Puncturing

The coding and puncturing section of the transmitter circuit is shown in Figure 23. It consists of a data scrambler (token 198) to randomize the incoming data, a convolutional encoder (token 7) and a puncturing circuit (token 8). The data scrambling is the result of the data exclusive or-ed (token 6) with a pseudorandom sequence produced by token 198.

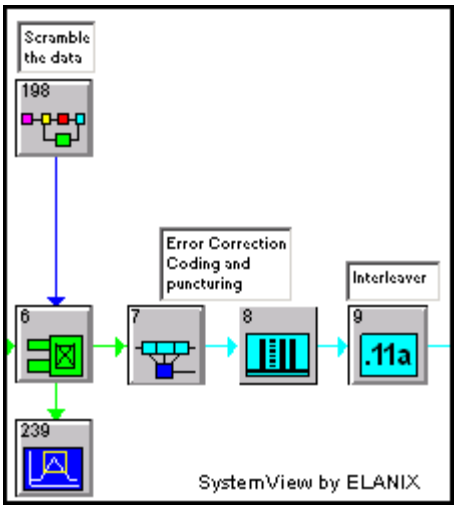

Figure 23 Coding and Puncturing Transmitter Circuit.

The pseudorandom generating MetaSystem (token 198) is shown in Figure 24. This circuit uses a seven-stage, 127-bit length, m-sequence generator with a random, non-zero initialization. Even though the communications token library of System View includes a PN generator token, the m-sequence generator used in this circuit was designed using individual flip-flops to allow for random initialization. The PN token in the source library cannot be used, since the length of the sequence cannot be specified.

The scrambler is made up of seven flip-flops from the System View logic library. The basic circuit design for the m-sequence generator is shown in Figure 25. This basic circuit is performed by tokens one through eight in the MetaSystem of Figure 24 in combination with token 6 in the transmitter circuit of Figure 23.

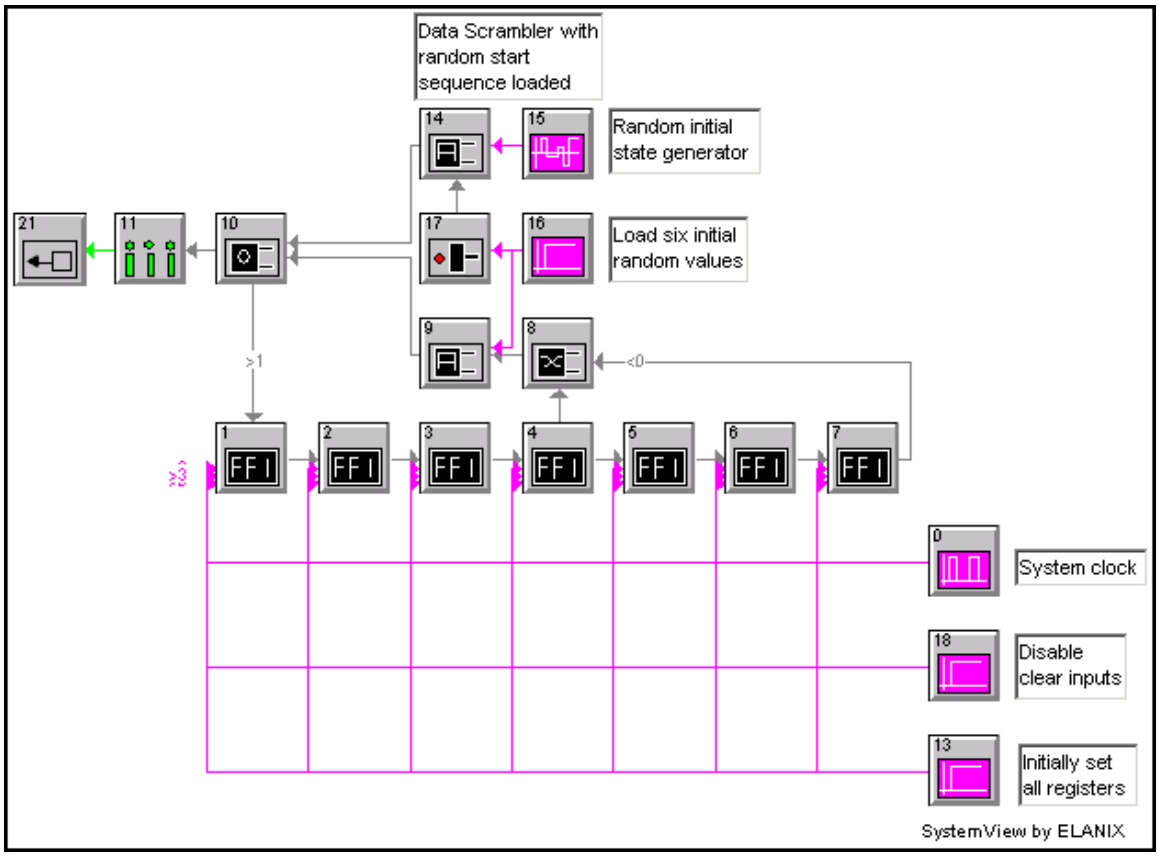

Figure 24 Data Scrambler MetaSystem.

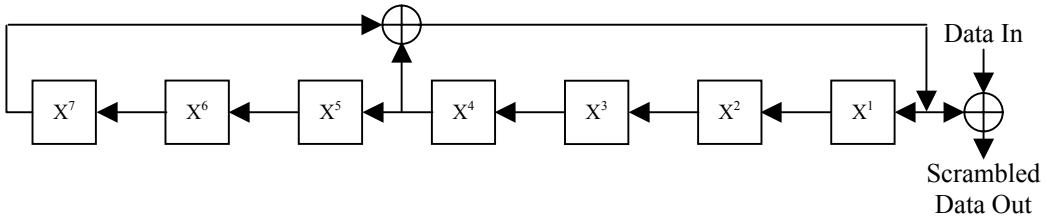

Figure 25 Data Scrambler [From Reference 10.]

The rest of the tokens in the MetaSystem are used to initialize the m-sequence generator and provide a clocking signal. MetaSystem token 16 provides an enabling signal for the initialization sequence. The step function start time is set for six clock cycles to allow the random sequence generated by token 15 to be shifted into the flip-flops. The flip-flops are initially all set to the logic one state to avoid the possibility of initializing all

the registers to zero. Once initialized, the AND gate (token 9) is enabled and the circuit works as shown in Figure 25.

Token 17 of Figure 24 is an inverter used to enable the AND gate (token 14) for the random sequence intializer. Token 10 is an OR gate used to let in either the initialization sequence or the feedback from the exclusive-or gate (token 8). Since the system clock is at 72 MHz and the data rate is at 36 MHz, the sampler (token 11) is used to step the scrambler sequence down to the data rate. Token 21 is the output token to connect the MetaSystem to the transmitter circuit.

Token 7 of the transmitter circuit (Figure 23) performs convolutional encoding of the scrambled data to support error correction in the receiver. The parameters used for this token are shown in Figure 26. The default encoder polynomials are automatically chosen by SystemView based on the coding rate (coding rate = information bits/output bits) and the constraint length. SystemView chooses the codes with the maximum free distance and, therefore, the best error performance capabilities. The polynomials can be changed by the user, if required. Since the encoder produces two output code bits for every input data bit, the data rate at the output of the encoder is 72 MHz. This illustrates the importance of knowing the clock rate at each point in the circuit, and it is one of the confusing aspects of using SystemView. Also, the encoder token inserts a two-sample delay in the output of the token.

Token 8 of the transmitter circuit (Figure 23) performs puncturing of the coded data bits to achieve the required code rate for the data rate being transmitted. As is shown in Table 3 of Chapter II, the required coding rate changes depending on the required data rate. This could be accomplished by using a separate encoder for each data rate, or it can be accomplished by using the same encoder for all data rates and then using puncturing to remove some of the coded bits to achieve the required coding rate. The Standard 802.11g specifies the later method. Specifically, the standard specifies the use of a rate $r = 1/2$ encoder flowed by puncturing, if required. The parameters used for the puncturing token are shown in Figure 27.

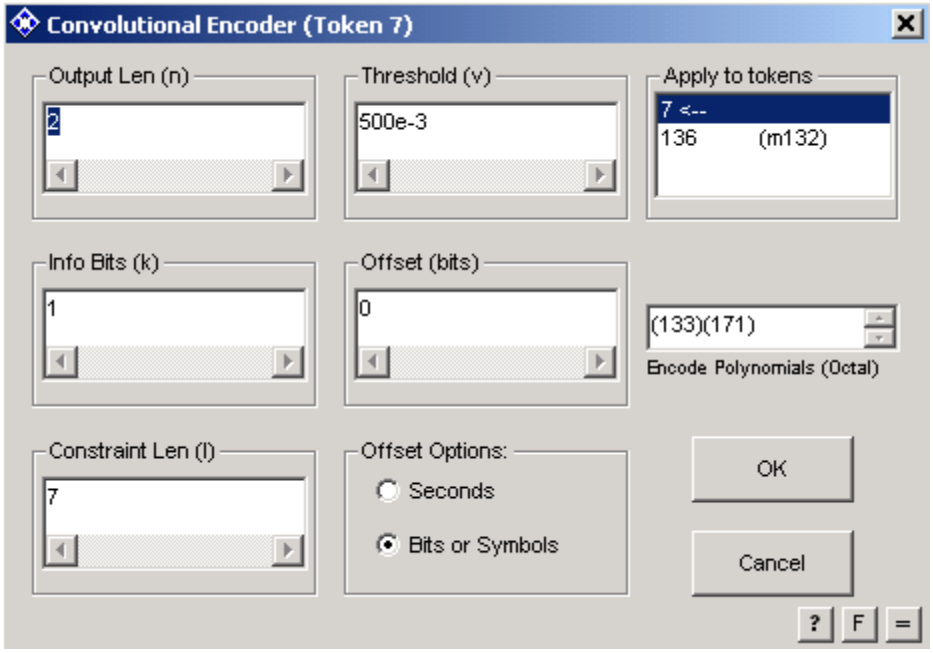

Figure 26 Convolutional Encoder Parameters.

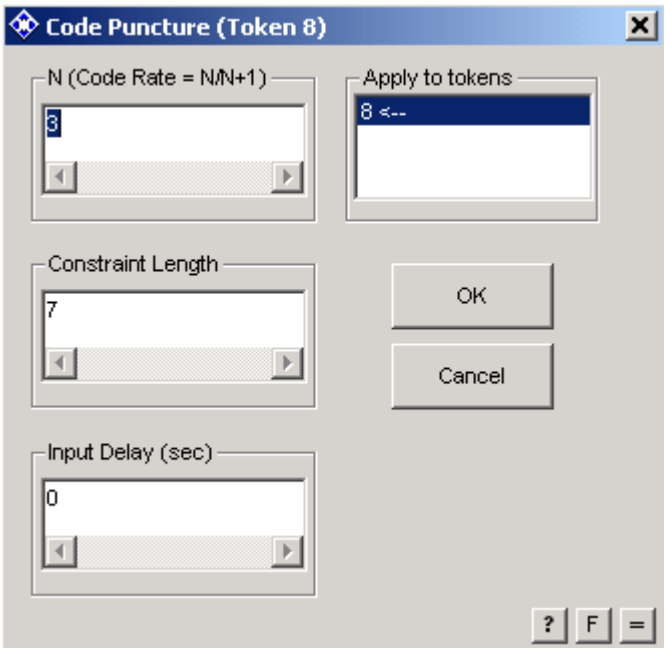

Figure 27 Puncturing Parameters.

As per Table 3, the required code rate is $r = 3/4$ which requires a value of $N = 3$. SystemView assumes the input to the puncturing token comes from a rate $r = 1/2$ convo-

lutional encoder. The constraint length of this encoder is the required parameter entry. Note again that the number of coded input bits is different than the number of coded output bits, since the token removes some of the coded bits. Since two out of every six bits of the input coded bit stream are removed, four out of every six coded bits remain. These four coded bits correspond to three input (pre-coding) bits; therefore the resulting code rate is $r = 3/4$. Therefore, the clock rate is reduced by a factor of $2/6 = 1/3$. The resulting output frequency is therefore $(2/3) \times (72 \text{ MHz}) = 48 \text{ MHz}$. The Standard 802.11g specifies that every fourth and fifth coded bit is discarded in every group of six coded bits to obtain the required $r = 3/4$ code rate. This token introduces a delay equal to six input data points at the output (i.e., the group of six inputs are read in before the group of four output bits are sent to the output).

Token 9 of Figure 23 performs the interleaving process in the transmitter. The input to token 9, as discussed above, is a 48-MHz unipolar encoded bit stream. The parameters used for this token are shown in Figure 28. The SystemView communications library originally only contained an interleaver capable of interleaving the 24-Mbps and the 36-Mbps data rates where the number of coded bits per symbol (N_{CBPS}) is 192. The SystemView wireless library contains an interleaver token that will correctly interleave all Standard 802.11a and Standard 802.11g OFDM data rates. However, later updates to System View changed the allowed data rates of the communications library interleaver to include all the allowable Standard 802.11a and Standard 802.11g data rates. The circuit of Figure 23 uses the communications library interleaver vice the wireless library interleaver. The interleaver inserts a 192-sample delay at the output. No interleaved data is read out until the entire block of 192 samples are read into the interleaver.

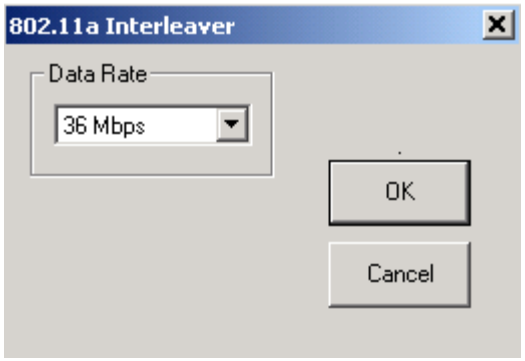

Figure 28 Interleaver Parameters.

3. Constellation Mapping

The 16-QAM constellation-mapping portion of the transmitter circuit is shown in Figure 29. It consists of a bit-to-symbol converter (token 10) and a QAM mapper (token 11).

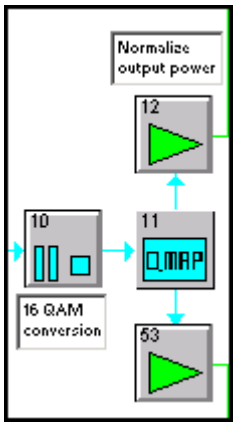

Figure 29 Transmitter Constellation Mapping.

The bit-to-symbol converter token converts a group of bits into an integer. In this circuit, a group of four bits is converted into integers from zero to fifteen. These integers are the required input for the QAM mapper token. The parameters used for this token are shown in Figure 30.

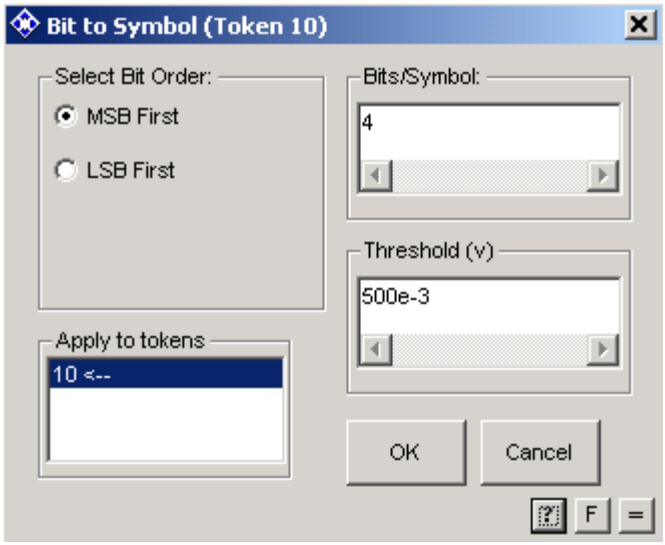

Figure 30 Transmitter Bit-to-Symbol Parameters.

The bit-to-symbol converter token allows the specification of the relative significance of the incoming bits. The requirement specified in the Standard 802.11a is that the first received bit is the most significant bit (MSB) and the last bit received is the least significant bit (LSB). Comparison to the threshold determines if the incoming bit is logic one (above the threshold) or logic zero (below the threshold). In this circuit, 0.5 volts is the threshold used. Note that this token reduces the data rate by a factor of four to 12 MHz, and it introduces a delay of four input samples, or one output sample. The output of this token cannot be displayed in the analysis window as voltage levels, since the output is an integer, not a voltage level. Changing the sink token receiving this token's output to receive numerical data will allow the output to be displayed.

The QAM mapper (token 11) of Figure 29 maps the integers provided by the bit-to-symbol converter (token 10) to in-phase and quadrature values specified by a mapping table. The parameters of the bit-to-symbol converter (token 10) are shown in Figure 30. The parameters used for the QAM mapper are shown in Figure 31. Since the transmitter circuit is using a data rate of 36 Mbps, the required QAM constellation size is sixteen, as specified in Table 3 in Chapter II. The required mapping is specified in Standard 802.11g and is shown in Table 4.

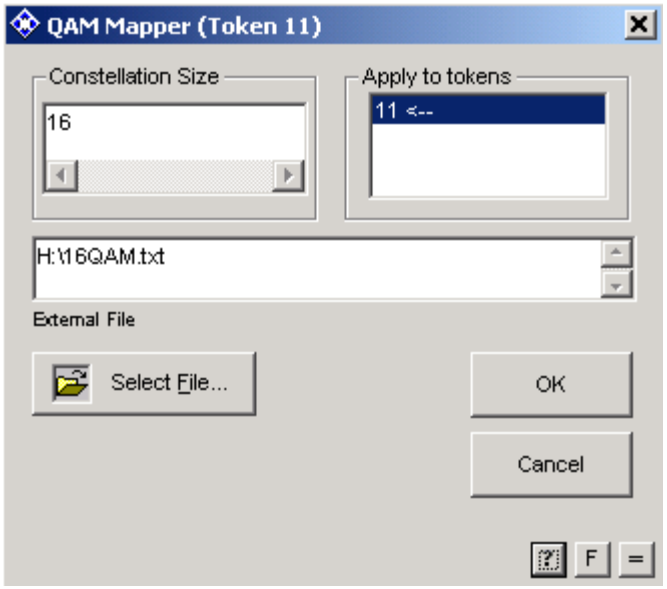

Figure 31 Transmitter QAM Mapper Parameters.

Table 4 16-QAM Encoding Table [After Reference 10.]

Input Symbol	In-Phase Output	Quadrature Output	Input Symbol	In-Phase Output	Quadrature Output
0	–3	–3	8	3	–3
1	–3	–1	9	3	–1
2	–3	3	10	3	3
3	–3	1	11	3	1
4	–1	–3	12	1	–3
5	–1	–1	13	1	–1
6	–1	3	14	1	3
7	–1	1	15	1	1

The file containing the mapping is specified in the parameter listing. No specific file format is shown in the documentation, however an example of the use of this token is provided in the example folder provided with the SystemView installation. It includes an

input file with the required format. The example table provided by SystemView was simply duplicated into the example file used in the simulation. The first line of the mapping file is ignored by the token and is provided as a header line. The second line of the file contains the number of constellation points. The third line contains labeling, and the rest of the lines contain the required mapping. The actual file used for this simulation is provided in Appendix A.

The gain tokens (token 12 and 53 in Figure 29) are used to normalize the power between the different mappings. The normalization factor achieves the same average power for all constellation mappings. For 16-QAM, this normalization factor is $1/\sqrt{10}$ which is used to multiply the amplitude of the coded data bits.

4. Pilot Tone Generation

Standard 802.11g adds pilot tones to each OFDM symbol to aid in maintaining the phase synchronization during the long packet transmission. If this did not occur, the carriers used in the OFDM symbol would slowly rotate out of phase, causing an error in the demodulation. The pilot tone is 250 kHz, which is added to each OFDM symbol in the form of pilot data spaced at an interval of 250 kHz.

The pilot tone generation circuit is shown in Figure 32. The pilot tones are placed in the data stream at IFFT input indices $n = -21$, $n = -7$, $n = 7$, and $n = 21$, with unity IFFT input value for indices $n = -21, -7,$ and 7 and minus one for $n = 21$. [10]

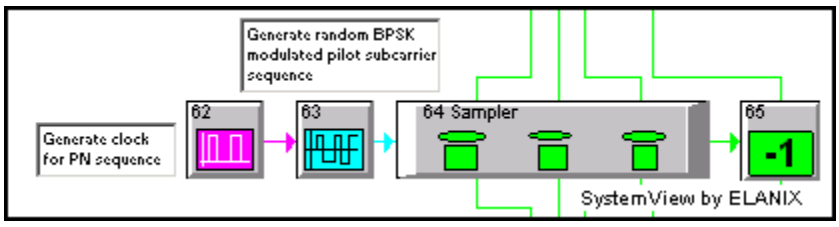

Figure 32 Transmitter Pilot Tone Generation Circuit.

To avoid the formation of spectral lines, the pilot symbol polarity is scrambled by the same pseudorandom sequence used to scramble the data. In this case, however, one

chip of the m-sequence scrambles all the pilot tone sequence values used in same OFDM symbol making the four pilot tone inputs to the IFFT $(1,1,1-1)$ or $(-1,-1,-1,1)$. Since the pilot sub-carriers are all ones (except the last one which is a minus one), the pilot tone sub-carriers can be generated with the random number generator at the symbol rate. The symbol interval specified by Standard 802.11g is $T = 4\ \mu s$, making the OFDM symbol rate $f = 250$ kHz. Token 62 of Figure 32 is a 250-kHz square wave clock for the PN sequence generator (token 63). Token 63 is a seven-stage shift register m-sequence generator, the same one as used in the data scrambler. The parameters used for token 63 are shown in Figure 33.

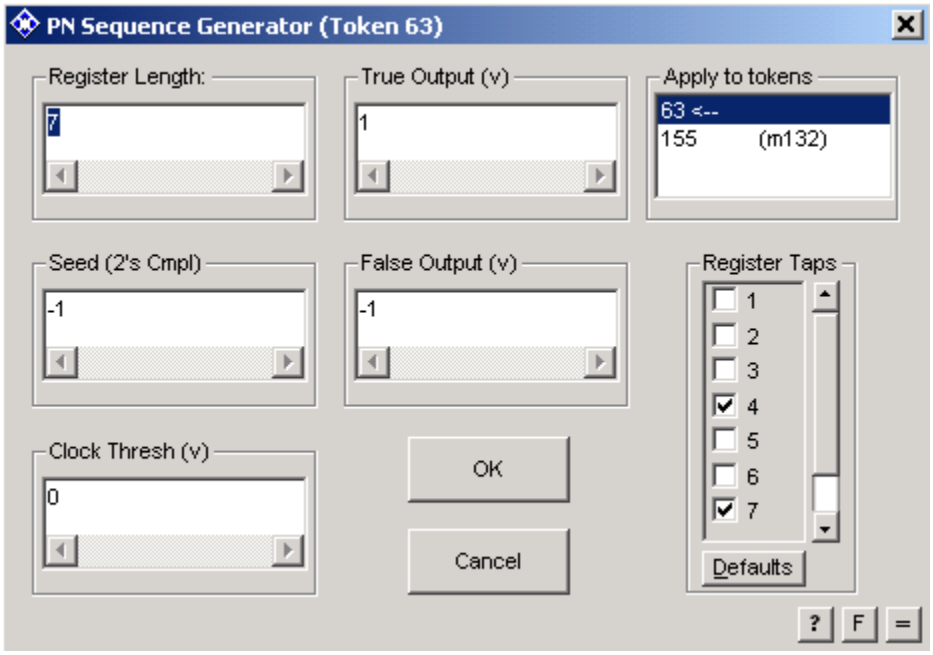

Figure 33 Transmitter PN Sequence Generator Parameters.

Note that the data produced is bipolar, non-return to zero with a true output at one volt and a false output at negative one volt. The clock is a square wave at ± 1 volt, allowing the clock threshold to be set at zero volts. The register taps come at the output of register four and register seven, as they did in the data scrambler.

The sampler (token 64) is used to bring the data rate down from the system rate of 72 MHz to the required rate of 250 kHz. The last of the four pilot sub-carriers is negated,

and all the sub-carriers are then multiplexed into the OFDM symbol. The pilot sub-carrier values for the in-phase and quadrature inputs to the IFFT are equal.

5. OFDM Symbol Formation

The OFDM symbols are formed by de-multiplexing the data symbols from the QAM mapper, inserting the pilot sub-carriers and the required zero sub-carriers to obtain a 64 sub-carrier OFDM symbol, and then multiplexing all the data together to input to the Inverse Fast Fourier Transformer (IFFT) used to modulate the data. The transmitter OFDM symbol formation circuit for the in-phase data is shown in Figure 34. An identical circuit is used for the quadrature data.

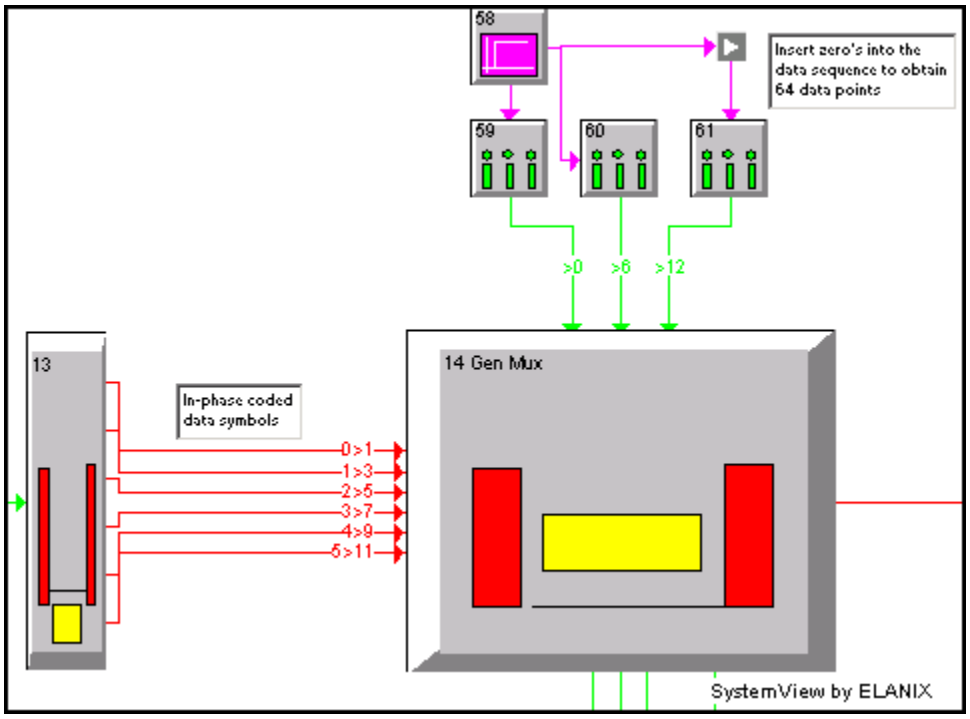

Figure 34 Transmitter OFDM Symbol Formation Circuit.

Standard 802.11g requires fifty-three data and pilot sub-carriers into the IFFT as shown in Table 5. Zeros are added on either side of the required sub-carrier allocation to make the IFFT symbol a full sixty-four sub-carriers long. Tokens 58 – 61 from Figure 34

insert these zeros into the data stream. Token 58 is a unit step function of zero and is the source of the zero sub-carriers.

Table 5 Sub-Carrier Frequency Allocation [After Reference 10.]

Sub-carrier Index	-32	-31	…	-27	-26	-25	…	-22	-21	-20	-19	…	-8	-7	-6	-6	…	-1	0
Sub-carrier value	0	0	0	0	d_0	d_1	…	d_4	P_0	d_5	d_6	…	d_{17}	P_1	d_{18}	d_{19}	…	d_{23}	0
Sub-carrier Index	0	1	2	…	6	7	8	…	20	21	22	…	25	26	27	28	29	30	31
Sub-carrier value	0	d_{24}	d_{25}	…	d_{29}	P_2	d_{30}	…	d_{42}	P_3	d_{43}	…	d_{46}	d_{47}	0	0	0	0	0

Sampler token 59 is used to produce the six zeros required before the beginning of the data stream. The parameters used are shown in Figure 35. The sample rate specified for the sampler is the rate required by the multiplexer to insert six out of sixty-four inputs at the symbol rate of 250 kHz. Since $6 \times 250 \text{ kHz} = 1.5 \text{ MHz}$, the required data rate into the multiplexer is 1.5 MHz.

The other required zeros are produced in the same manner. Sampler token 60 is used to insert the one zero at the carrier frequency (zero sub-carrier), and sampler token 61 is used to insert five zeros after the data sequence at a frequency of $5 \times 250 \text{ kHz} = 1.25 \text{ MHz}$.

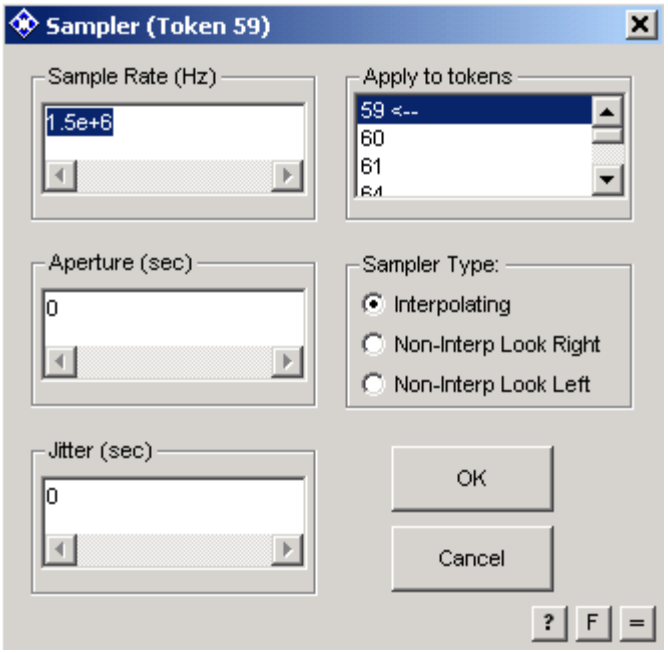

Figure 35 Transmitter Zero Insertion Sampler Token Parameters.

The de-multiplexer (token 13) is used to separate the data into the sequences specified in Table 5. The parameters used are shown in Figure 36.

The coded data symbols are split up into six segments as specified in Standard 802.11g. The slot boundary offset input parameter for the de-multiplexing token is used to start the de-multiplexing after a certain number of received data samples. In this circuit, the slot boundary is left at the default value of zero. As mentioned before, the de-multiplexer token introduces a delay equal to the number of input samples, and it changes the data rate of the samples at each one of its outputs. The output rate is given by

$$\text{Rate} = \frac{\#\text{ of input samples}}{\text{Total Frame Duration}}. \tag{3.1}$$

For the first segment of data ($d_0 - d_4$), the number of samples specified for the token input is five, and the total frame length is $48/\left(12\times10^{6}\right) = 4\ \mu\text{s}$. Equation (3.1) gives us a data rate of $5/\left(4\times10^{-6}\right) = 1.25$ MHz for the first output of the de-multiplexer. The other output data rates can be calculated in the same manner.

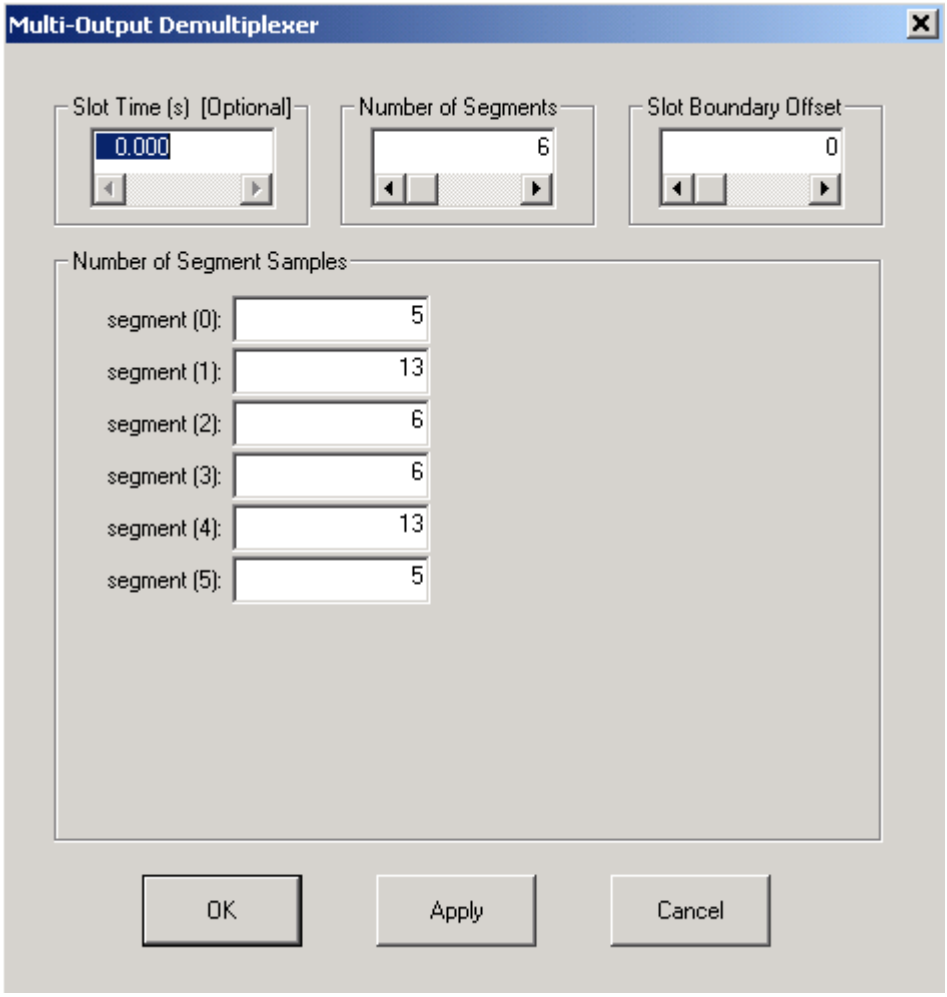

Figure 36 Transmitter De-Multiplexer Parameters.

The general multiplexer (token 14) is used to put the entire OFDM symbol together. The parameters of this token are shown in Figure 37. A total of thirteen inputs are multiplexed together to form the OFDM symbol. With all the inputs coming in at different rates, it is easy to get the incorrect data rates and cause an error during execution of the system. Although not specifically written in the documentation for SystemView, the incoming data must satisfy (3.1) and the following equivalent rate

$$\text{Rate} = \frac{\#\text{ of input samples}}{\text{Total Samples} * \text{Output Rate}}. \tag{3.2}$$

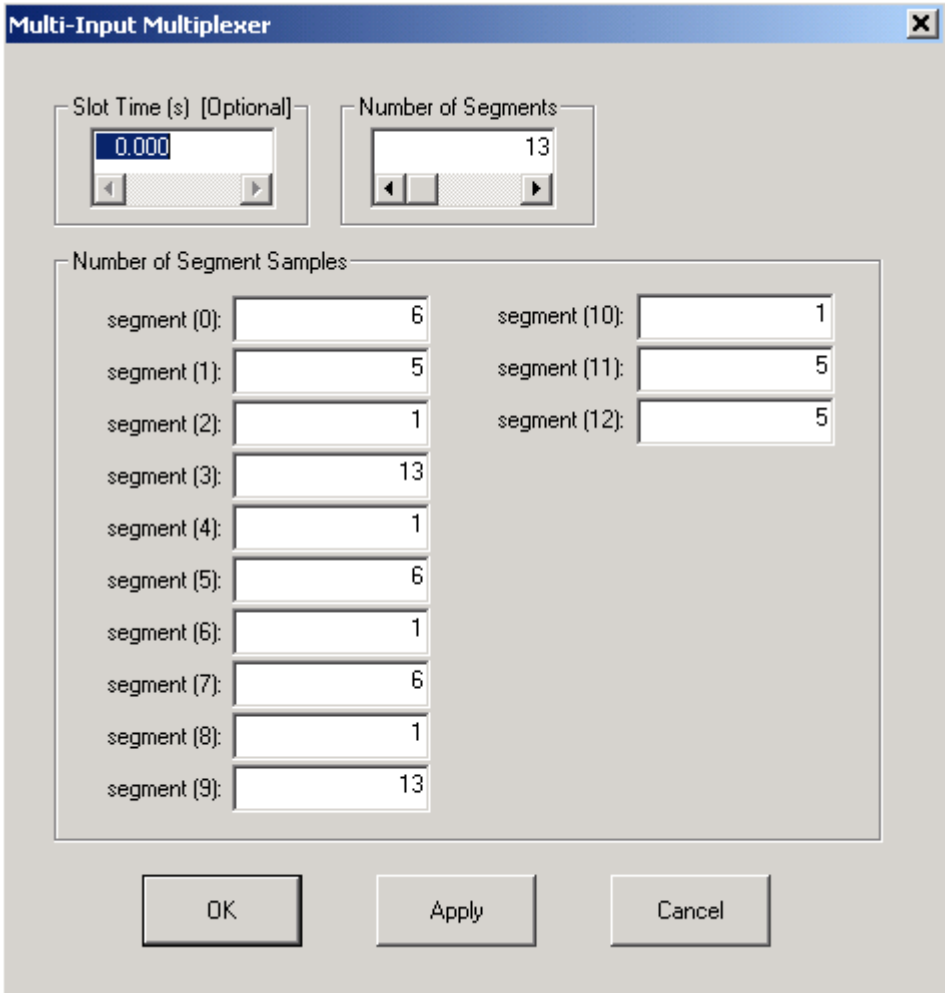

Figure 37 Transmitter Multiplexer Parameters.

Equation (3.2) is the easier one to use in this case since we know that the required output data rate must be $64/\left(4\times10^{-6}\right)=16$ MHz. The input rates can then be checked to ensure they meet the requirements of (3.2). Note that the data rate changed from 12 MHz at the input to the de-multiplexer and multiplexer portion of the circuit to 16 MHz at the output. The multiplexer also delays the samples by the time required to gather all the inputs.

6. IFFT Modulation

The IFFT modulation of the coded data is shown in Figure 38. The data is modulated with an IFFT of the OFDM modulator (token 17). This token takes the IFFT of the

sixty-four data points and cyclically extends it by the required guard interval. The cyclic extension is placed at the beginning of the sixty-four-point output, as required by Standard 802.11g. The inputs are the complex coded data symbols, and the outputs are complex also, requiring real and imaginary data streams.

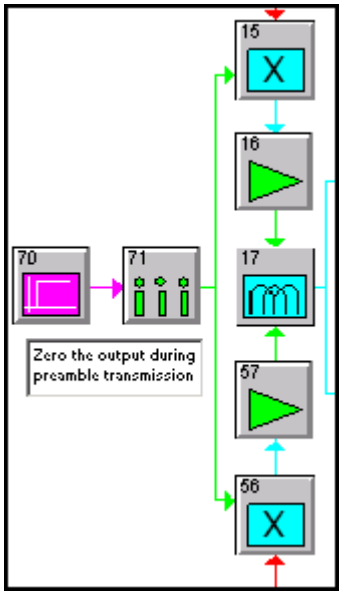

Figure 38 Transmitter IFFT Modulation Circuit.

The parameters used in the OFDM modulator token are shown in Figure 39. The symbol time and the guard interval are specified by Standard 802.11g [11]. The token has sixty-four inputs and $64+16=80$ outputs in the symbol interval of 4 μs. This gives an output data rate of $80/\left(4\times10^{-6}\right)=20$ MHz. However, since the token output is not exactly as required by Standard 802.11g, corrections to the output data stream are made during the windowing process following the modulation.

The remainder of the tokens used in the IFFT modulation portion of the circuit are required to zero the output during the transmission of the preamble and header of the OFDM packet. The unit step function (token 70) is used to enable the multiplexer output after the transmission of the header and preamble. The time to transmit the short training symbols is 8 μs and the time to transmit the long training symbols is also 8 μs for a total

of 16 μs per Standard 802.11g [11]. The unit step function is enabled at 12 μs , accounting for the 4 μs delay of the OFDM modulator token. The unit step function is changed to the required data rate of 16 MHz and multiplies the output from the multiplexer as a window for the outgoing data stream. The signal into the OFDM token is divided by the number of data points, $N = 64$, by the gain tokens (tokens 16 and 57) to match the test data output used for comparison in Appendix G of Reference 10. The normalization is not required by Standard 802.11g but is customary in the applications using the IFFT and FFT.

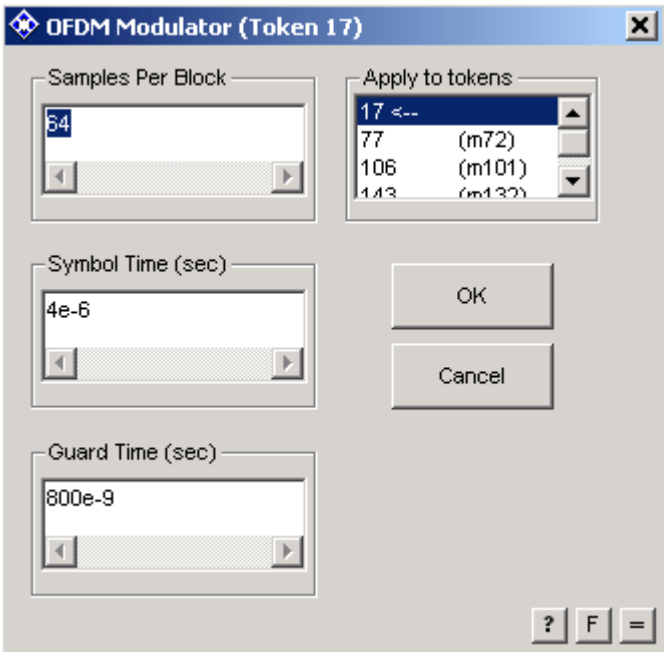

Figure 39 OFDM Modulator Parameters.

7. Preamble and Header Generation

The preamble and header portions of the transmitter circuit are shown in Figure 40. The generation of the short and long training symbols is done with the MetaSystem token 84. This is added to the windowed OFDM data symbols to produce the OFDM packet. Tokens 18 and 43 window the in-phase and quadrature signals, respectively.

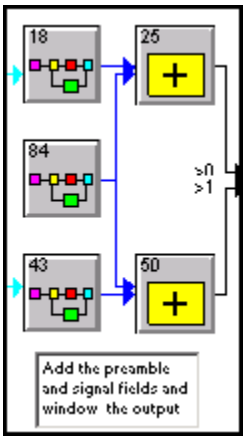

Figure 40 OFDM Preamble and Header Generation Circuit.

The token 84 MetaSystem is shown in Figure 41. The preamble consisting of the long and short training symbols is formed by the MetaSystem token 0. The header consisting of the signal field is formed by the MetaSystem token 240. The service field portion of the header is formed and modulated with the data. These two data fields are simply added together to form the preamble and the header. Since the output is formed with quadrature and in-phase data, two channels are required.

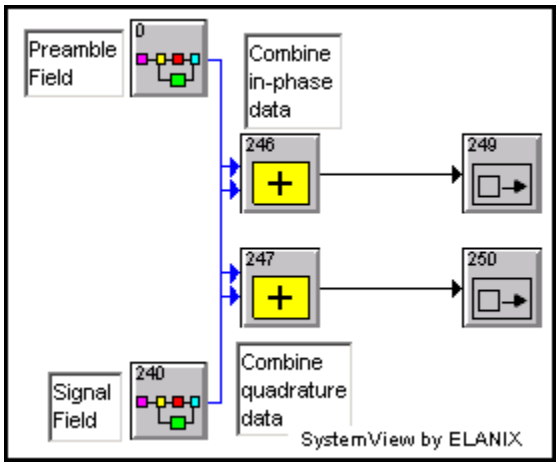

Figure 41 Packet Preamble and Header Generation Circuit.

The preamble MetaSystem (token 0) is shown in Figure 42. The MetaSystem of token 185 forms the short training symbols, and the MetaSystem of token 204 forms the

long training symbols. Since the short training symbols are transmitted first in the packet, the start of the long training symbols must be delayed by $8\ \mu s$, the length of the short training symbols. Again, two channels are used to form training symbols for both the quadrature and in-phase data streams. The parameters for the sample delay (token 234) are shown in Figure 43. The parameters for token 235 are the same. The sample delay is not equal to the total length of the short training symbols, since the long training symbols take longer to generate. The sample delay was adjusted until the last short training symbol overlapped with the first long training symbol, in this case 144 samples.

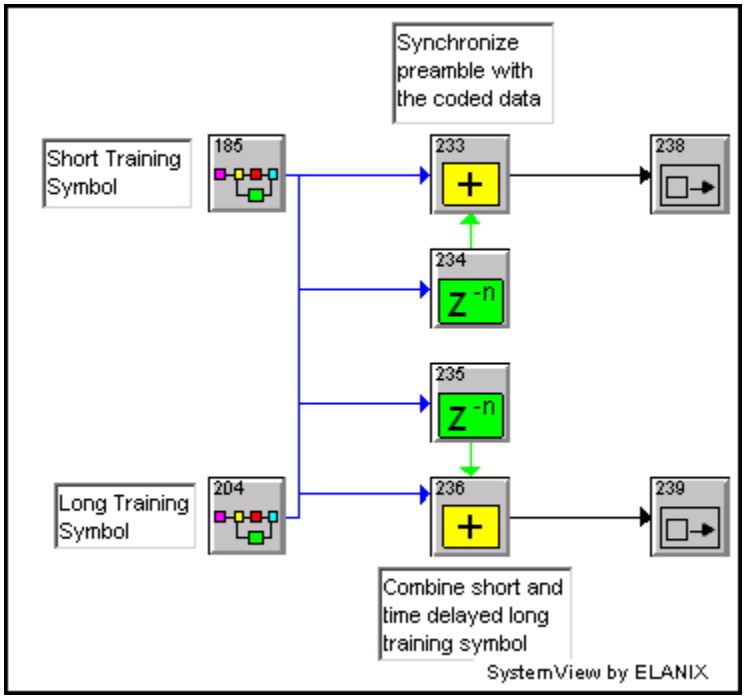

Figure 42 Preamble Generation Circuit.

The short training symbol generation circuit is shown in Figure 44. The short training symbols are generated by the short training sequence given by $S = \{0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 0, 0, 0, 0, -1-j, 0, 0, 0, -1-j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0, 0, 1+j, 0, 0\}$. This sequence is formed by an impulse function (token 0) and linear filter (token 2) with an impulse response equal to the required data sequence.

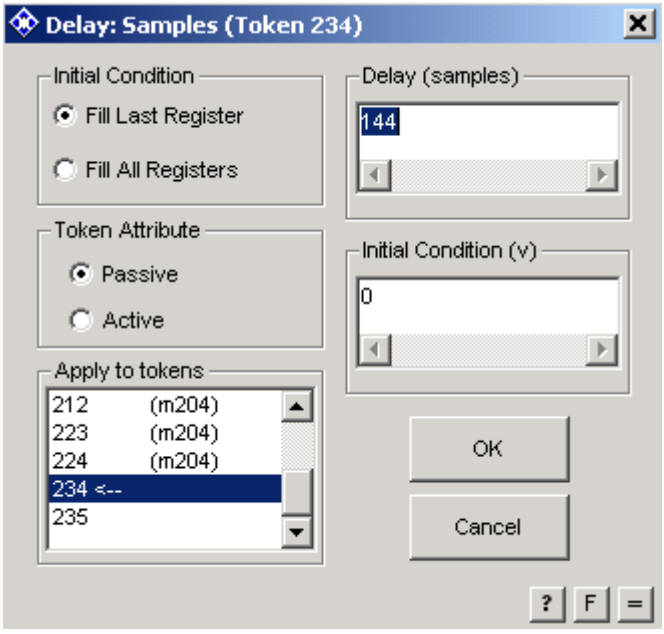

Figure 43 Transmitter Sample Delay Parameters.

The parameters for the impulse source (token 0) are shown in Figure 45. The parameters for the linear filter (token 2) are shown in Figure 46. The impulse source token generates a single pulse of width equal to the system clock period and amplitude of the inverse of the system clock period, producing a pulse with unit area. This pulse is attenuated to unit amplitude by the gain input parameter. The linear filter produces the 64 inputs for the OFDM modulator (token 13). Since the real and imaginary input values to the modulator are the same, one filter can drive both inputs. The desired values are simply entered as the numerator z-coefficients of the desired impulse response. Although not realistic in practice to generate, it provides a quick and easy way to generate a desired sequence. This could be implemented in a real circuit with a ROM chip that has the required sequence stored in memory. Since the purpose of this thesis is to test different receiver designs, the transmitter just needs to generate the required signal, not be built using real circuit components.

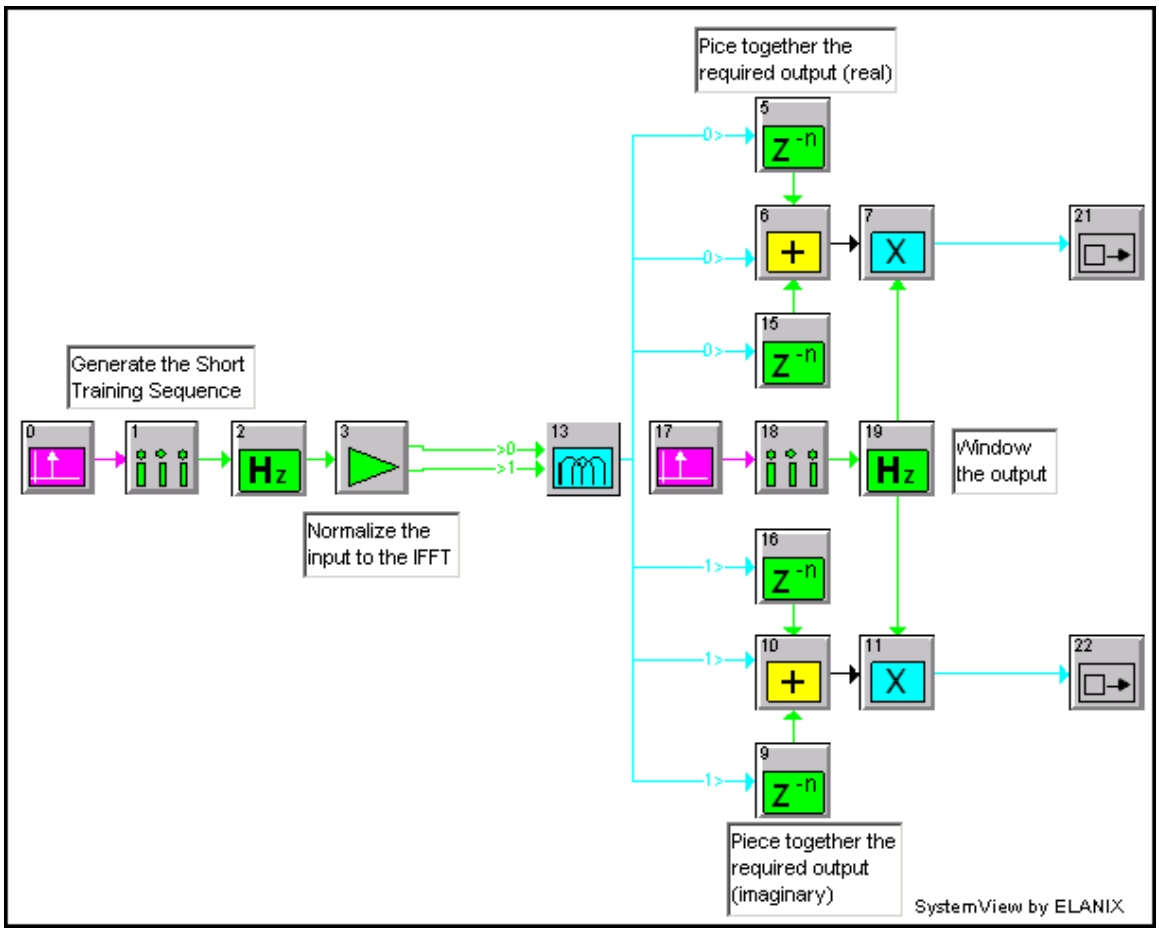

Figure 44 Short Training Sequence Generator Circuit.

The output of the linear filter is normalized by dividing by $N = 64$ with the gain token 3, the number of data points used in the IFFT of the OFDM modulator (token 13). The remainder of the short symbol circuit is used to window the short symbol and overcome a bug in the OFDM modulator token that prevents cyclically extending the output beyond half the IFFT length. Tokens $17-19$ are used to produce the required window in the same way that the short sequence was generated by tokens zero through two. The window in this case multiplies the first and last data points by 0.5 and does not affect the other data points. This is the recommended windowing to limit the spectrum of the transmitted signal. Tokens 5, 6 and 15 for the in-phase data and 9, 10, and 16 for the quadrature data piece together the entire short data symbol.

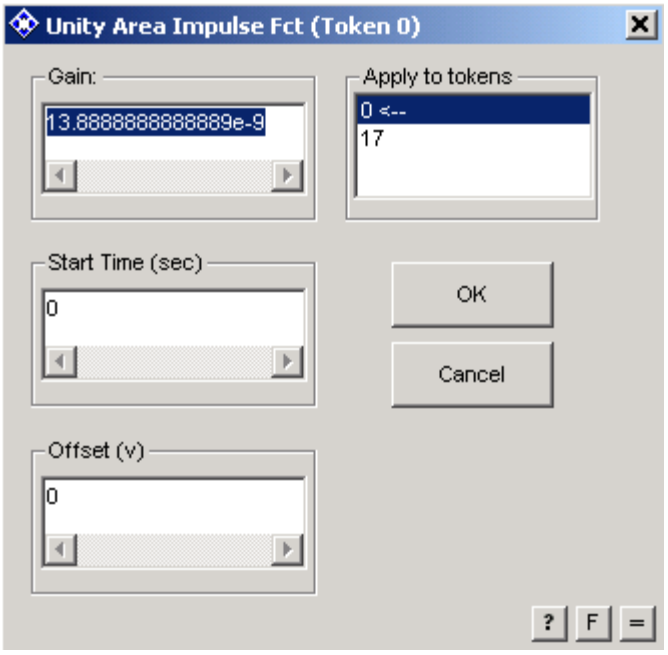

Figure 45 Short Symbol Impulse Parameters.

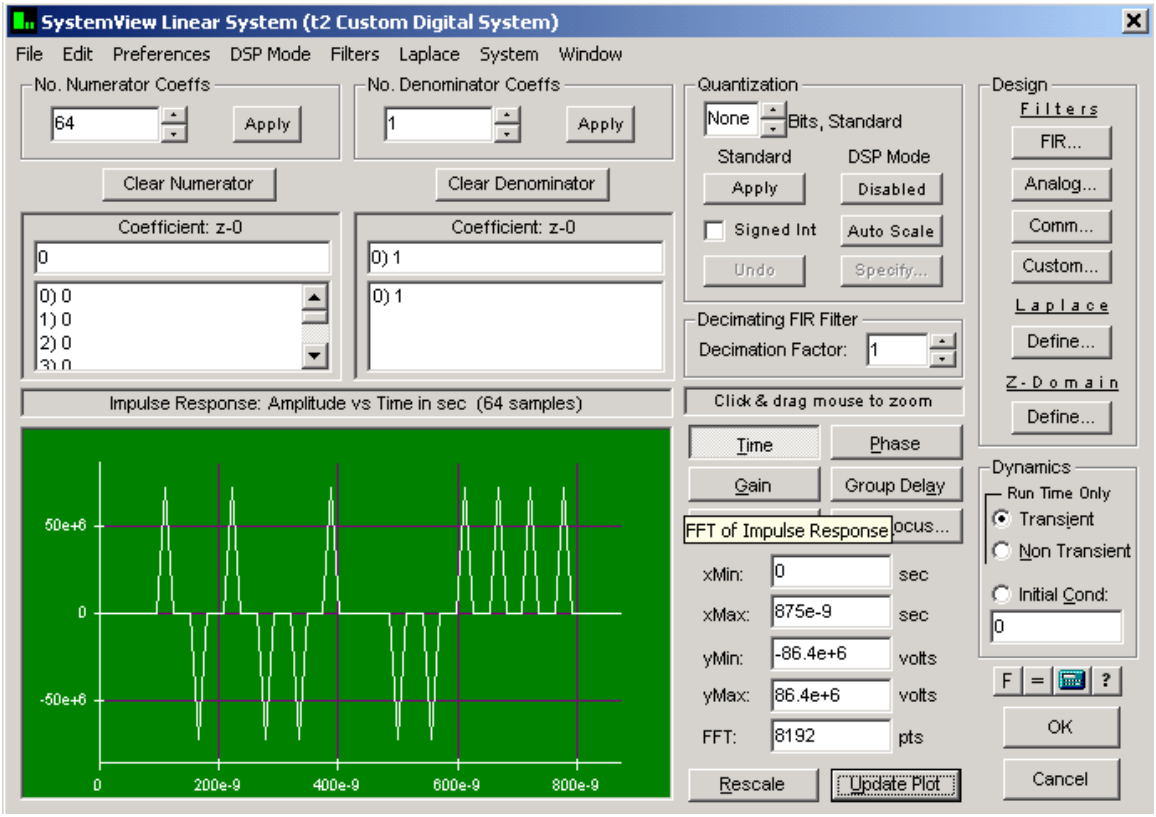

Figure 46 Short Training Symbol Linear Filter Parameters.

The OFDM modulator (token 13) parameters are shown in Figure 47. No guard interval is specified, since cutting and pasting the output of the token form the required symbol. The output is added to a sixty-four-sample delay of the output (token 15), which is added to a 128-sample delay (token 5). The output of the adder is the cyclic repetition of the OFDM modulator output extended over three periods. Since only two and one-half periods are required, the result is windowed to provide the desired signal. The window size is 161 samples, and the required short symbol preamble is shown in Figure 48. Note that there is a delay of 4 μs before the output of the short symbol due to the inherent delay of the OFDM modulator token. The output of the short training symbol MetaSystem matches the short symbol data generated in Table G.4 of Reference 10.

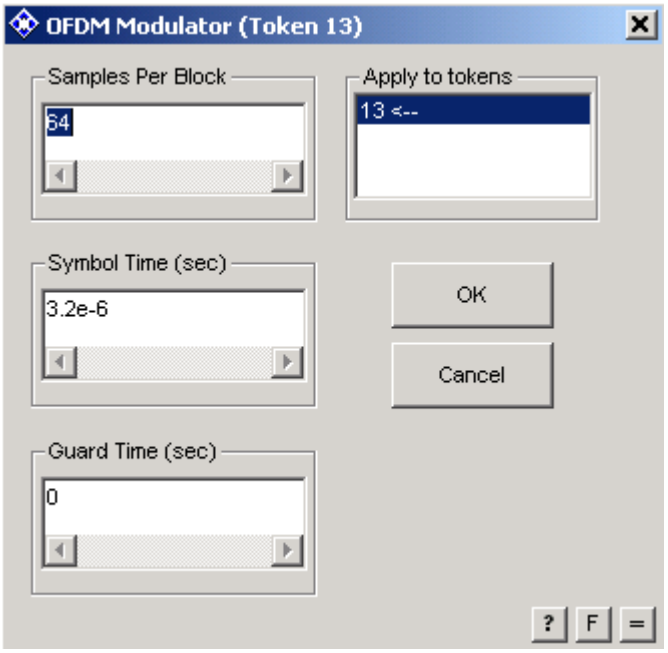

Figure 47 Short Symbol OFDM Modulator Parameters.

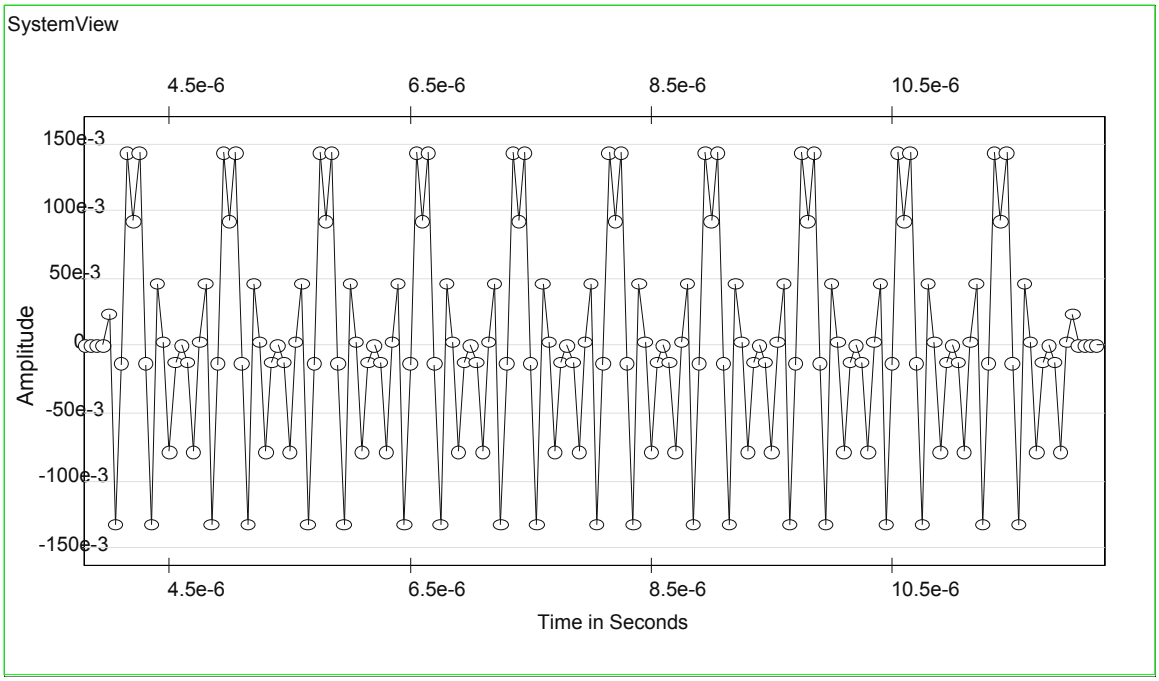

Figure 48 Short Symbol Training (In-Phase Data).

The long training symbol MetaSystem, token 204 of Figure 42, is shown in Figure 49. It is generated in the same manner as the short training symbol: an impulse drives a linear filter with an impulse response equal to the desired training sequence.

The long training sequence is given by $L = \{1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 0, 1, -1, -1, 1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1\}$. Note that the values are all real, requiring a zero imaginary value input to the OFDM modulator token. These values are produced by the linear filter (token 2) of Figure 49, normalized by dividing by $N = 64$, and fed into the real input of the OFDM modulator. The parameters used for this OFDM modulator (token 4) are shown in Figure 50. The data leaving the OFDM modulator has the correct guard interval of 1.6 μs , but only half (3.2 μs) of the long training symbol. The tokens forming the inputs of the adders (tokens 7 and 12) of Figure 49 cut and paste the OFDM modulator output to obtain the required long training symbol. The step functions (tokens 20 and 26) are used to filter out unwanted data form the output of the modulator. The sample delays (tokens 6, 11, 16, and 18) delay by 64 and 128 samples to place the data in the required position. The remainder of the tokens in Figure 49 window the

output as was accomplished in the short training symbol generation circuit. The resulting long training symbol is shown in Figure 51.

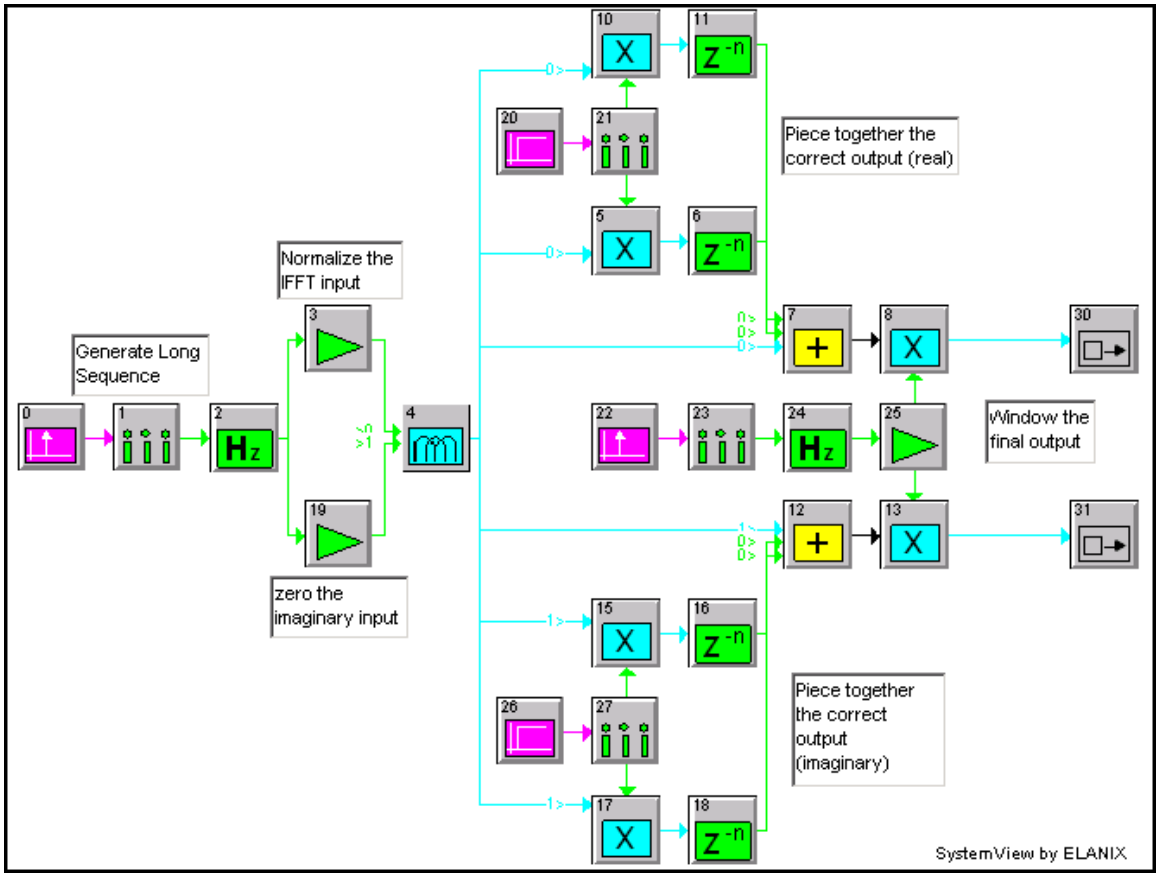

Figure 49 Long Training Symbol Generation Circuit.

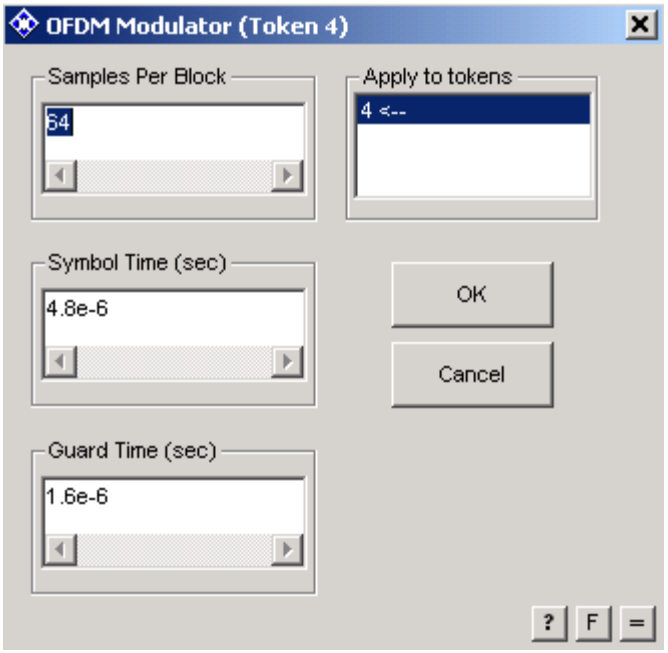

Figure 50 Long Training Symbol OFDM Modulator Parameters.

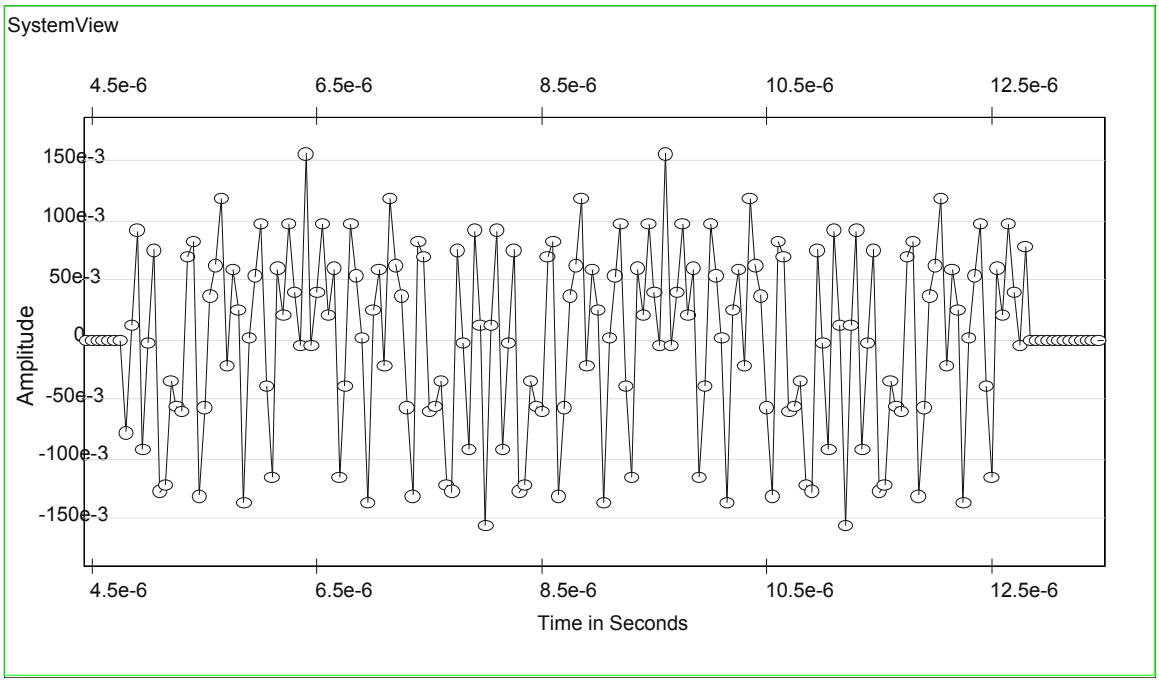

Figure 51 Long Training Symbol (In-Phase Data).

This data output from the long training symbol generation circuit matches the long training symbol generated in Table G.6 of Reference [10].

The signal field MetaSystem (token 240) of Figure 41 is shown in Figure 52 and is similar in structure to the transmitter circuit of Figure 15. This is due to the fact that the header is transmitted using the 6-Mbps data rate. Since the rest of the OFDM packet is sent using the 36-Mbps data rate, a separate circuit must be used to support the generation of the header.

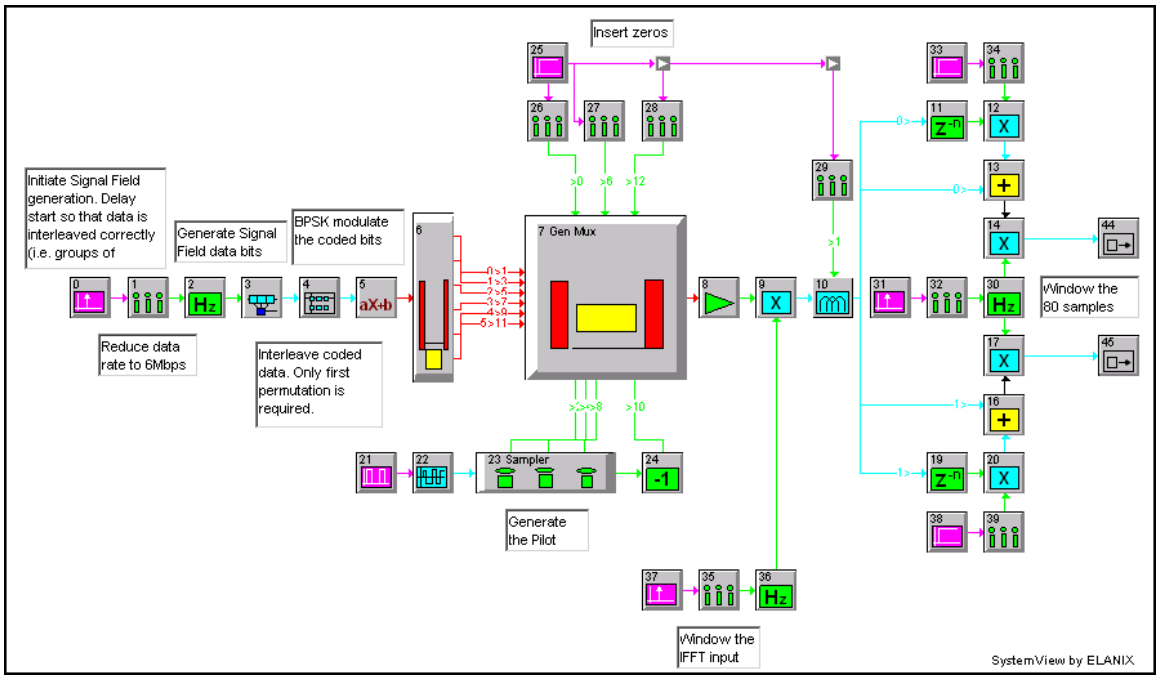

Figure 52 Signal Field Generation Circuit.

The service field format is shown in Figure 53. With the rate for this circuit chosen as 36 Mbps, the length chosen as the maximum of 4095 octets, the parity bit is chosen to obtain even parity, and the tail bits are used to reset the transmitter circuit, the signal field sequence is $S = \{1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$ [10]. This sequence is produced in the same manner as the required sequences for the short and long training symbols with the unit impulse source, the sampler, and the linear filter combination (tokens 0, 1, and 2 of Figure 52). The sequence is encoded with the same rate $r = 1/2$ convolutional encoder as is used in the transmitter circuit of Figure 15. Since the data rate is 6 Mbps, the coding rate out of the encoder is at the required coding rate of Table 3 and no puncturing is required. The interleaver (token 4) is not the same

one as used in the transmitter circuit, since the number of coded bits per symbol is $N_{CBPS} = 48$ for the 6-Mbps data rate, and the Standard 802.11g interleaver was originally designed to work only with the 36-Mbps data rate. For the 6-Mbps data rate only the first stage of the Standard 802.11g interleaver is required, which is a straightforward block interleaver. The parameters for this interleaver are shown in Figure 54. Sixteen columns is the required block length per Reference [10], and since $N_{CBPS} = 48$, then this results in $48/16 = 3$ rows used in the interleaver.

Rate 4 bits	Reserved 1 bit	Length 12 bits	Parity 1 bit	Tail 6 bits

Figure 53 Service Field Format.

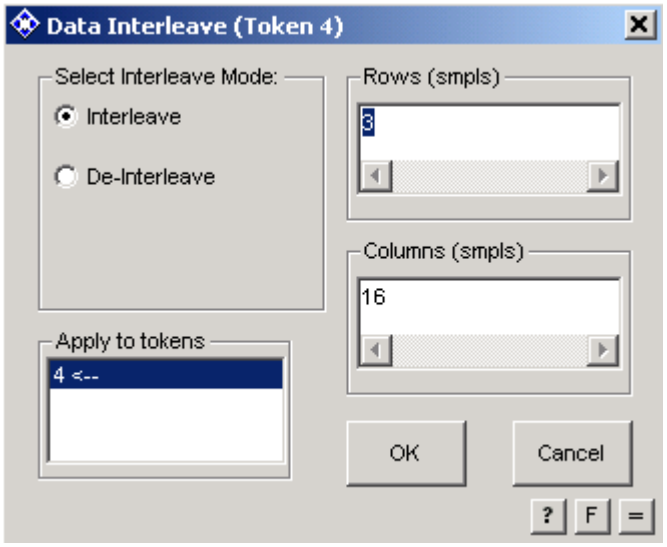

Figure 54 Signal Field Interleaver Parameters.

After interleaving, the data is binary phase-shift keying (BPSK) modulated by the polynomial token (token 5). The parameters for this token are shown in Figure 55. This token does a linear shift of the one and zero volt inputs to produce a one and minus one output. The parameters specify the equation $output = 2 \times input - 1$.

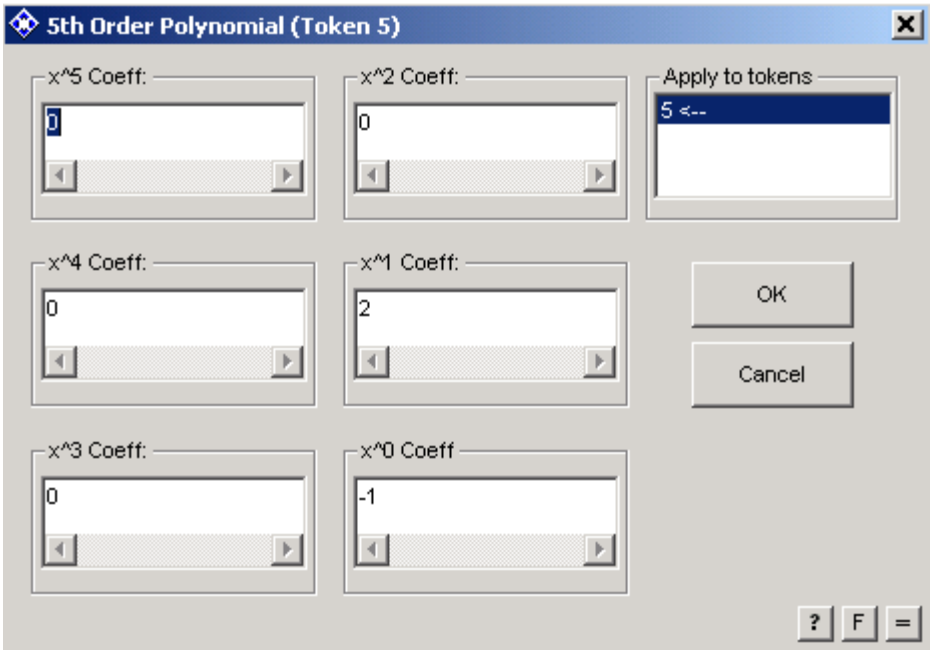

Figure 55 Signal Field BPSK Modulator Parameters.

After BPSK modulation, the pilot symbols and zeros are added to obtain sixty-four inputs for the IFFT operation of token 10 in the same manner as the transmitter circuit of Figure 15. The output of the IFFT is then windowed in the same fashion as the transmitter circuit.

Tokens 18 and 43 of the transmitter circuit of Figure 15 are used to window each OFDM data symbol generated by the transmitter. Since this windowing process must occur on a continuous basis, the one time windowing used in the preamble and header generation will not work and a different technique of windowing is needed. Both windowing tokens are identical. The MetaSystem of token 18 is shown in Figure 56.

The window function is [10]

$$w_T[n] = \begin{cases} 1 & 1 \le n \le 79 \\ 0.5 & n = 0, n = 80. \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The last ($n = 80$) data point of (3.3) is added to the first ($n = 0$) data point of the next symbol.

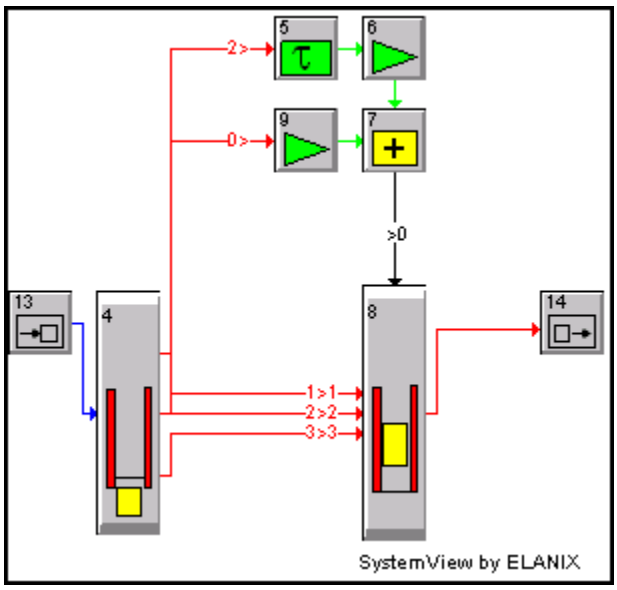

Figure 56 OFDM Data Symbol Windowing Circuit.

The de-multiplexer (token 4) parameters are shown in Figure 57. This de-multiplexing allows the first number of the OFDM symbol sequence, representing the first value in the guard interval, and the first value of the original IFFT without extension to be removed from the data stream and modified for windowing.

The first bit in the OFDM symbol sequence is multiplied by 0.5, as shown in (3.3), by the gain token. This value is added to the last value of the previous symbol by delaying the de-multiplexer output two by $4\ \mu s$ (the length of one OFDM symbol) and dividing it by two. The composite signal is then recombined by the multiplexer (token 8).

This windowing technique also modifies the OFDM modulator token output so it produces the correct Standard 802.11g OFDM symbol. In effect, it takes the first data point after the guard interval and puts a copy of it at the end of the symbol. This produces a total of 81 ($n = 0$ to $n = 80$) data points, and follows the packet formation example in Appendix G of Reference [10]. The first and last data points of the sample are multiplied by 0.5 and added to the previous and subsequent OFDM symbols.

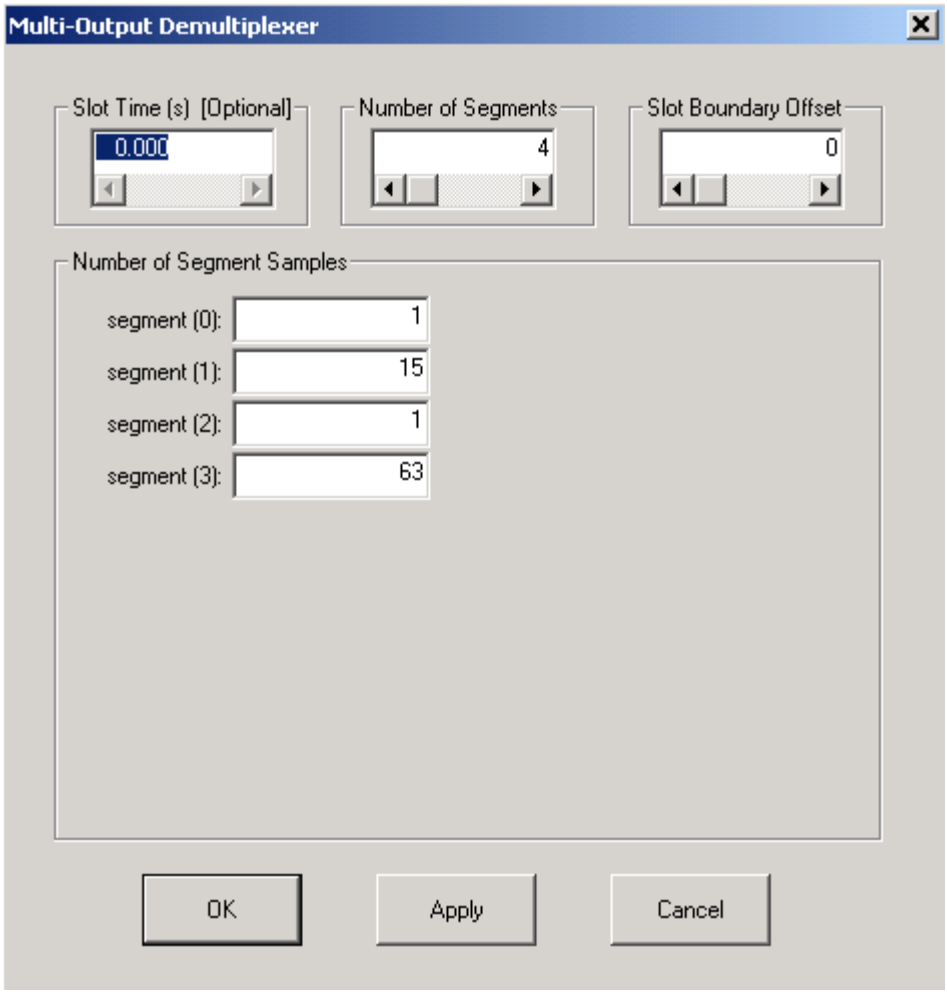

Figure 57 Transmitter OFDM Data Symbol Windowing De-multiplexer Parameters.

An example output of the transmitter is shown in Figure 58. It shows the short and long training sequences and the first data OFDM symbol. This is the in-phase data from the output of the transmitter. The output has a delay of one OFDM symbol (4 μs) that is due to the inherent delay of the OFDM modulator used in the generation of the short training symbol.

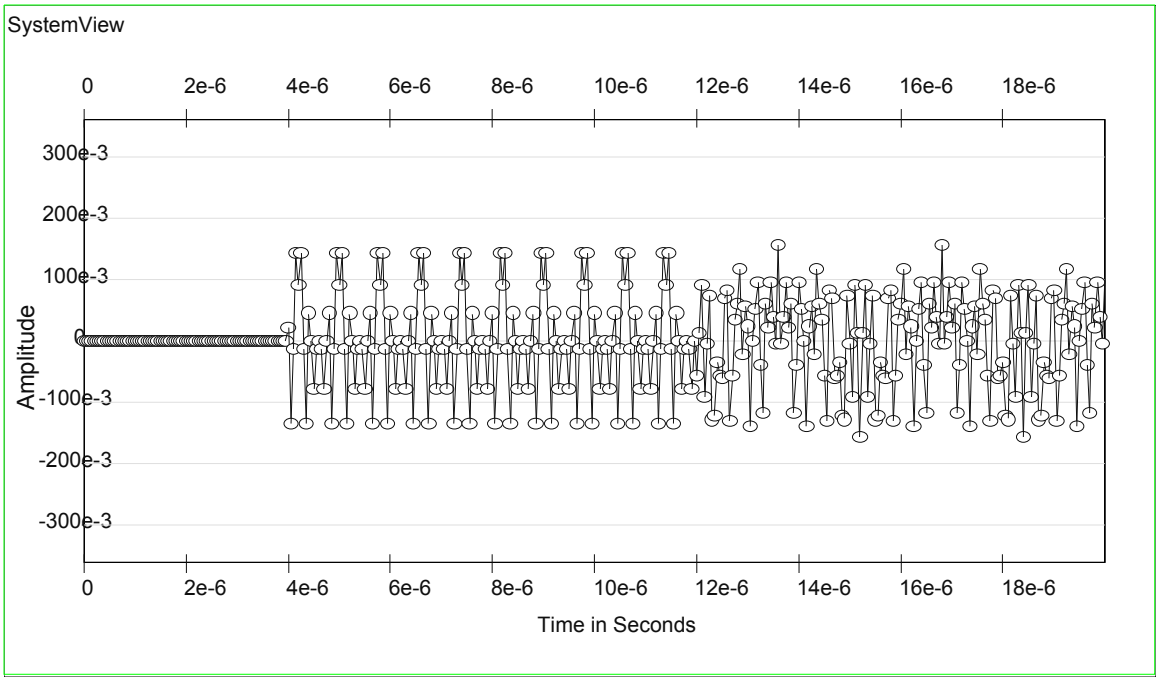

Figure 58 Transmitter Output.

C. RECEIVER

The basic OFDM receiver used in this thesis is shown in Figure 59. Since no synchronization circuitry is required, the design is relatively simple.

The basic functions of the receiver are OFDM demodulation, stripping off the header information, de-modulating the 16-QAM symbols, de-interleaving, de-puncturing, and decoding the received signal to recover the original scrambled data bits. This signal is then descrambled and the output is compared to the generated data input to obtain a bit error rate (BER). With no noise input into the system, this receiver combined with the OFDM transmitter of Section B produces no bit errors, validating the proper demodulation of the OFDM data. Several system runs using 2^{20} samples, able to detect bit error rates of 10^{-5}, returned no errors. Additional runs at 2^{24} samples and 2^{27} samples, able to detect bit error rates of 10^{-6} and 10^{-7} were also run with no errors in the system.

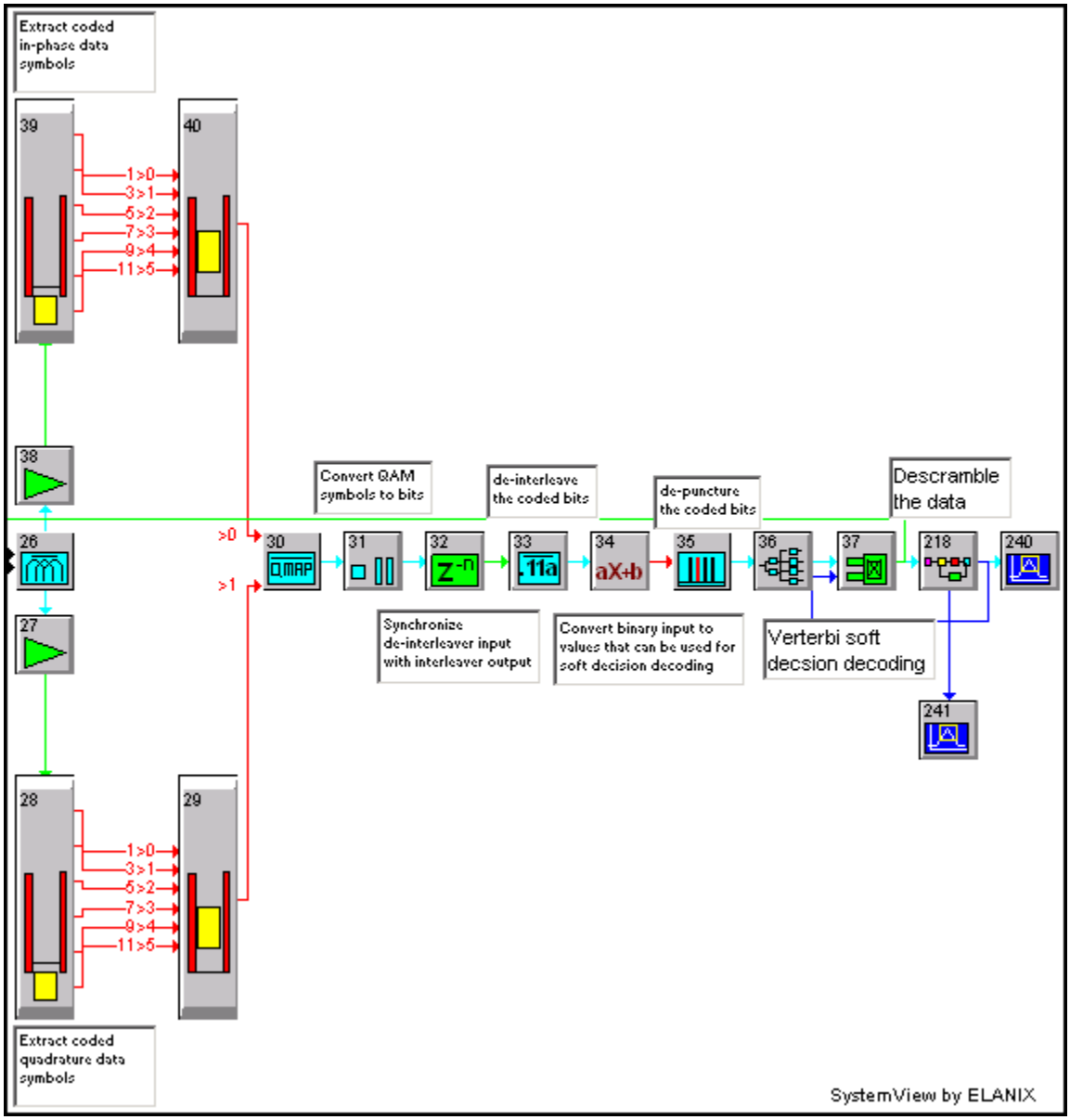

Figure 59 OFDM Basic Receiver Circuit.

1. FFT Demodulation

The OFDM demodulator (token 26) parameters are shown in Figure 60. The 24-µs input delay strips off the 4-µs transmitter delay, the short training symbol (8 µs), the long training symbol (8 µs), and the OFDM symbol containing the signal data (4 µs). The OFDM demodulator strips off the data contained in the guard interval of the received data and performs an FFT on the remaining 64 data points. The output clock

frequency is reduced by a factor of $64/80 = 4/5$ due to the number of input data points being equal to 80 and the number of output data points being equal to 64. Since the input frequency is 20 MHz, the token output frequency is $(4/5) \times 20 \text{ MHz} = 16 \text{ MHz}$. The output is delayed by another $4 \mu s$ while the input data is being read into the token.

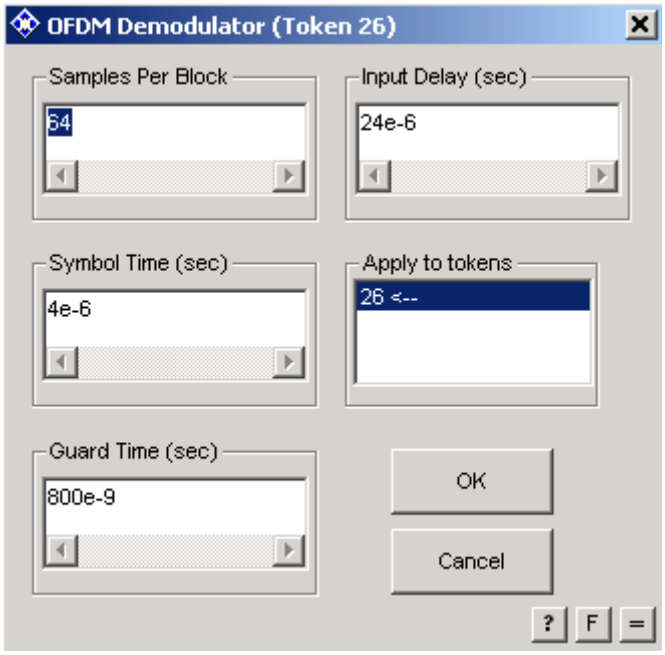

Figure 60 OFDM Demodulator Parameters.

2. Data Extraction

The circuit of Figure 61 performs the removal of the extra zeros and the pilot tones. The de-multiplexer divides the data up into the thirteen segments used in the transmitter's multiplexer to divide the incoming data stream into the desired data bits and the undesired zeros and pilot tones. The five segments from the de-mulitplexer that contain the desired data are fed into the multiplexer. The other outputs from the de-multiplexer are not used. This is the reverse process performed by the multiplexer and de-mulitplexer in the transmitter.

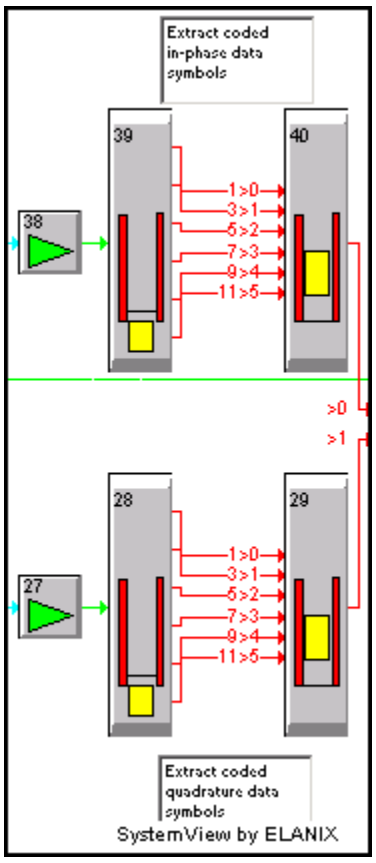

Figure 61 Receiver Data Extraction Circuit.

3. Hard Decision Demodulation

The demodulation section of the receiver is shown in Figure 62.

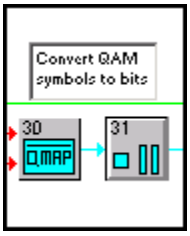

Figure 62 Receiver Data Demodulation Circuit.

Demodulation consists of mapping the in-phase and quadrature data onto the constellation provided by an input file, in this case the same one used in the transmitter circuit shown in the Appendix A. The parameters used in the QAM de-mapper are shown in

Figure 63. Since the 36-Mbps data rate uses 16-QAM modulation, the constellation size is 16. The output of the QAM de-mapper is an integer corresponding to the closest I and Q in the constellation.

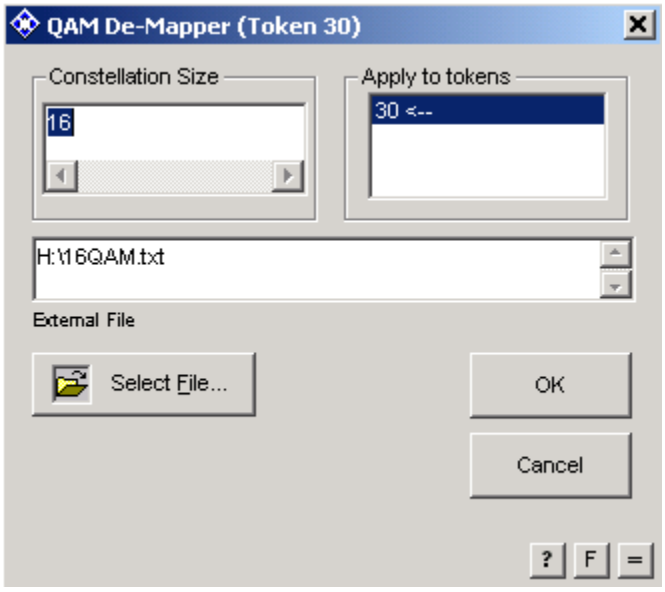

Figure 63 Receiver QAM De-Mapper Parameters.

The symbol-to-bit converter (token 31) takes the integer provided by the de-mapper and converts it to a stream of four bits. The parameters for the converter are shown in Figure 64. The ordering of the bits is as specified in Reference [10].

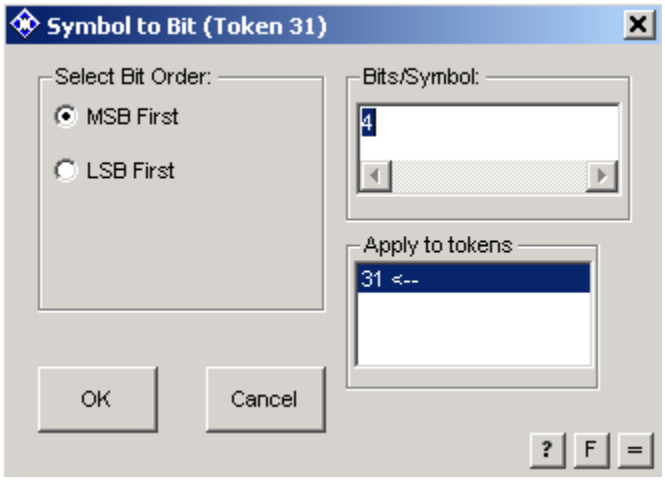

Figure 64 Receiver Symbol-to-Bit Converter Parameters.

4. De-Interleaving and De-Puncturing

Once the incoming data has been converted back to binary digits, the interleaving and puncturing operations performed in the transmitter must be reversed. Figure 65 shows the circuit used to perform this process. The first operation preformed is the de-interleaving by token 33. Since this token begins de-interleaving upon commencement of the data run, a sample delay token must be used to ensure the received data enters the de-interleaver at the beginning of a 192 sample de-interleaving process. To compute the required delay, the individual delays from each token can be calculated, or the cross-correlation of the interleaver output and the de-interleaver input can be performed. The sample delay with the greatest correlation will be the required sample delay at the input to the de-interleaver. Performing this operation on this circuit results in a maximum correlation at 1156 samples. Since $1156 = 6 \times 192 + 4$, the de-interleaver input is delayed by $192 - 4 = 188$ samples. This means the data will reach the input of the de-interleaver at the start of the seventh de-interleaving cycle. Any change in the signal path between the interleaver and de-interleaver will require a recalculation of this delay.

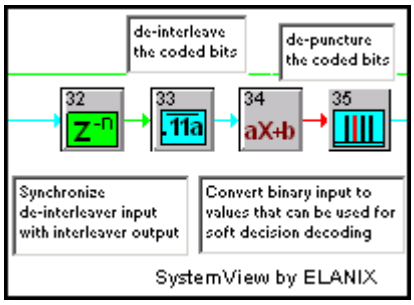

Figure 65 De-Coding and De-Puncturing Circuit.

One the data has been de-interleaved, it must be scaled to allow the de-puncture operation to insert a zero (or null value) wherever data was removed from during the puncturing operation. The equation used in the polynomial token is $output = 2 \times input - 1$, the same as was used for BPSK modulation of the signal field in the transmitter.

The parameters used in de-puncturing (token 35) are shown in Figure 66. The parameters are the same as entered for the puncturing operation, except the input delay. In order to decode correctly, the punctured data points must be inserted into the correct positions in the data stream. In this circuit, the input delay was also used to zero out the noise produced from the QAM demodulation process. With zero input, QAM demodulation produces an alternating output. To ensure the decoder started in the zero state when the data first reaches it, the de-puncturing is delayed until the first data arrives (coded zeros). Doing this, however, can make placing the inserted bits tricky. It could also be ignored, since enough bits are used in the transmitter to zero out the decoder prior to real data arriving by design. However, this circuit was designed to produce a zero output from the decoder until the data arrives.

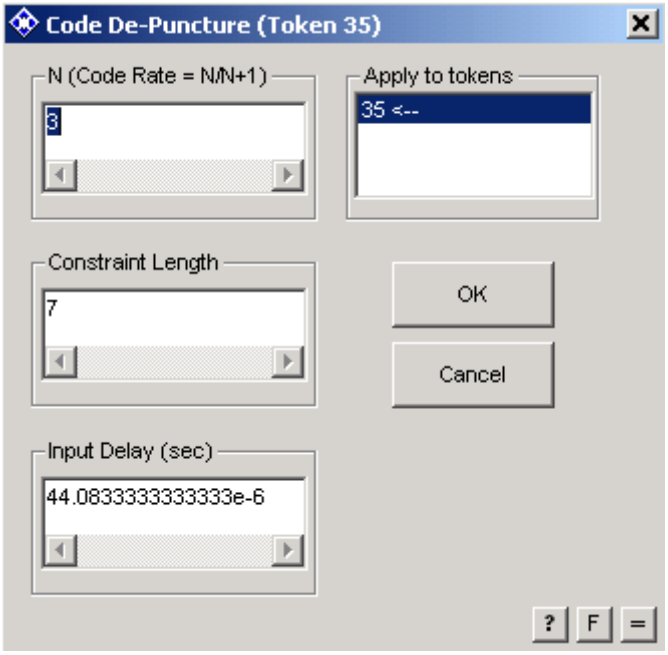

Figure 66 De-Puncture Parameters.

5. Viterbi Decoding

The decoding process uses a soft decision Viterbi decoder. The soft decision decoding allows the insert bits from the de-puncture operation to be weighted at a value of something other than a zero or one. However, the complete decoding process is hard decision, since the value of the input symbol has already been evaluated during the QAM de-mapping process of token 30. The circuit used for decoding and de-scrambling the data is shown in Figure 67.

The parameters for the decoder (token 36) are shown in Figure 68. For the most part, the parameters are the same as the convolutional encoder used in the transmitter circuit. The path length chosen is the same used in Reference [15]. Although for this circuit, with no channel or system noise, a path length of one worked just as well. With channel noise, the best path length would be 28 (4×7). The parameters chosen for the soft decision decoding follow those recommended in Reference 15.

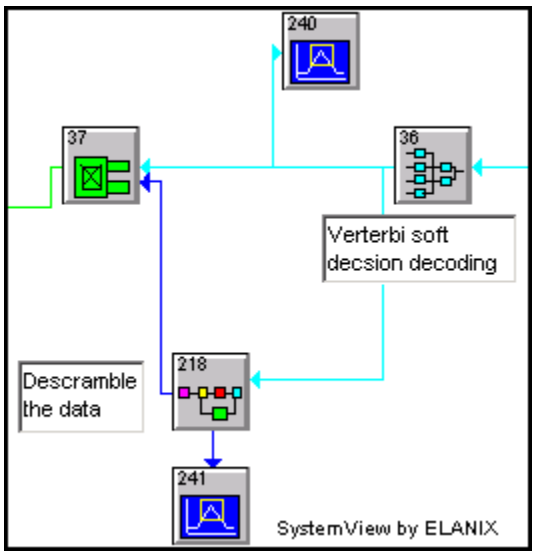

Figure 67 Decoding and De-scrambling Circuit.

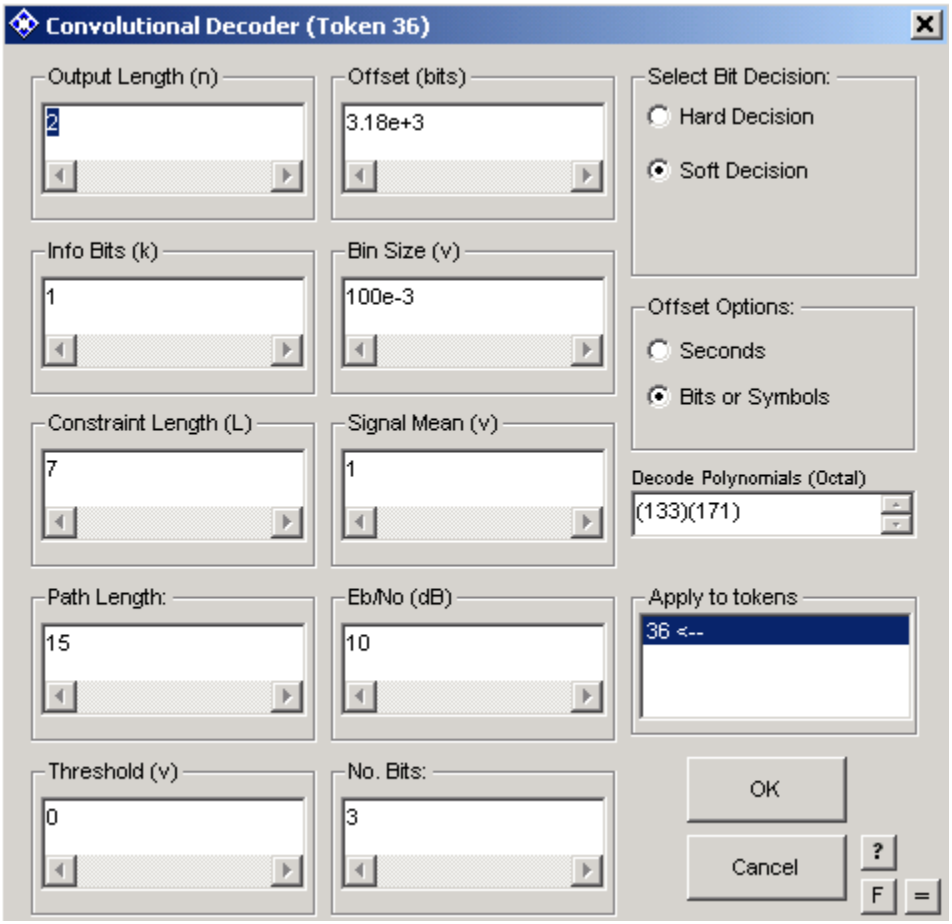

Figure 68 Decoder Parameters.

The offset is used to zero the output until the transmitted data is received. A correlation between the convolutional encoder output and the decoder input was used to calculate the sample delay required.

The descrambler is shown in Figure 69. The descrambler is the same basic circuit as was used in the scrambler in the transmitter circuit. The same seven-stage m-sequence generating circuit is used. The difference is that the data from the decoder is run through the flip-flops until the seven bits from the service field are entered as an initialization for the sequence. The step function then turns off the data input from the decoder and the m-sequence generator runs on its own, decoding the incoming data by exclusive-oring the sequence with the received data.

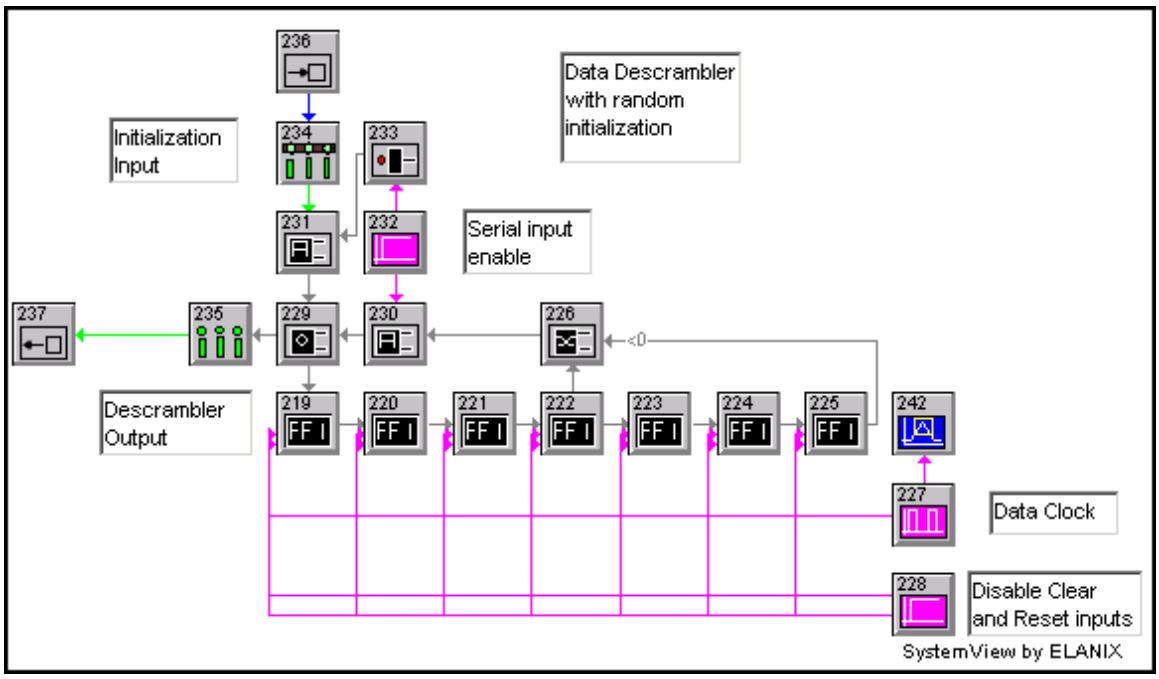

Figure 69 Data De-Scrambling Circuit.

D. BIT ERROR RATE CALCULATIONS

The circuit to calculate the bit error rate in the receiver is shown in Figure 70. Token 4 is the bit error rate counter token from the communications library. It takes two inputs and compares binary values over a certain number of trials. The parameters for the BER token are shown in Figure 71. The number of trial bits is set to one so that bits are

compared one by one. The threshold is the value that separates logic one from logic zero. The offset is used to ignore initial data, which is set to zero in this case. The sample delay token is used to delay the original data the same amount as the receiver and transmitter circuit so that initial data bits transmitted are compared with initial data bits received. As in the de-puncture and de-interleave operations, the easiest way to obtain this value is to use the sink calculator in the System View Analysis window to calculate the correlation between the data output from the PN generator in the transmitter and the data output from the decoder in the receiver. The time delay of maximum correlation is the desired sample value. The value obtained in this circuit was 1607 samples.

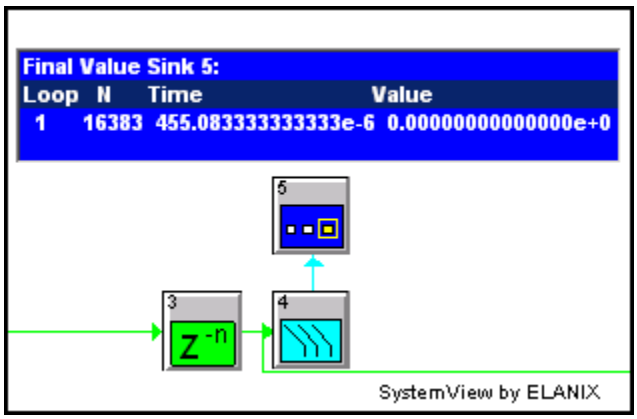

Figure 70 Bit Error Rate Calculation.

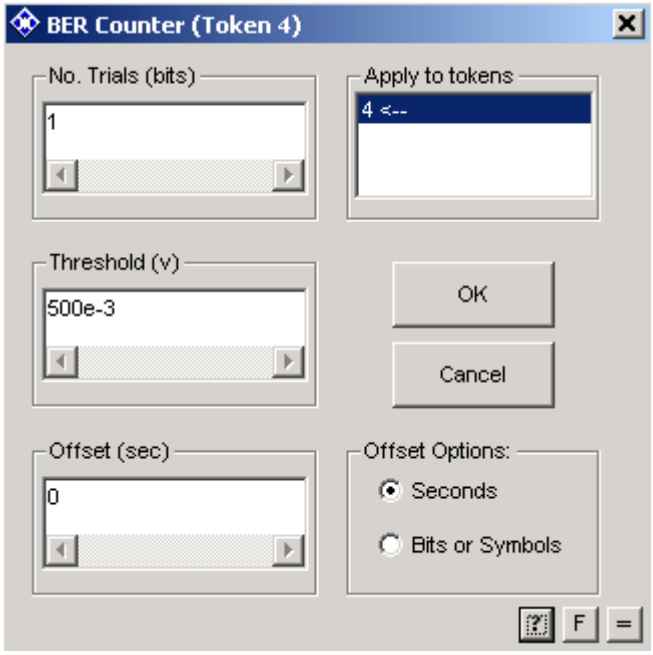

Figure 71 Bit Error Rate Parameters.

Although the receiver presented in this section is very basic, it serves as a basis for building receiver circuits that will optimize synchronization of an OFDM signal. Small modifications of the receiver are made to accommodate the various synchronization techniques.

IV. OFDM SYNCHRONIZATION

There are many aspects to synchronizing the receiver with the transmitted signal in an OFDM system. In addition, in a connectionless, packet-switched communications system such as Standard 802.11g compliant systems, the receiver has very little time to synchronize with the transmitter. Even with a packet length of 4096 bytes, the maximum allowed, the entire packet could be at most 627 μs at a data rate of 54 Mbps. The preamble and the header are a total of 20 μs, leaving little time to achieve synchronization before the start of the data. Since the header contains information required for proper demodulation of the OFDM signal, synchronization must be accomplished in the time allotted for the long and short training sequences, which is 16 μs. Once synchronization is achieved, it must be maintained over the duration of the OFDM symbol. If the channel characteristics change significantly during the transmission of the packet, synchronization and the data can be lost, resulting in retransmission and an overall slower throughput.

The synchronization of the OFDM signal is broken down into four different steps. The first is packet detection, or the ability of the receiver to detect all the incoming packets and reject all the spurious signals caused by noise or other communications operating in the same frequency band. The second is received frequency synchronization, or ensuring that the receiver local oscillator frequency is the same as the carrier frequency of the received signal. The third is the carrier phase-offset synchronization, or ensuring that the receiver local oscillator phase is the same as the carrier phase of the received signal. The last is OFDM symbol synchronization, or the sampling of the data symbols at the correct time and entering the correct data into the FFT to demodulate the OFDM signal, which includes synchronizing the data clocks between the receiver and transmitter.

The communication signals that follow the 802.11 wireless standards contain several segments meant to enhance and aid synchronization. In particular, Standard 802.11g uses the short and long training symbols for synchronization as illustrated in Figure 72.

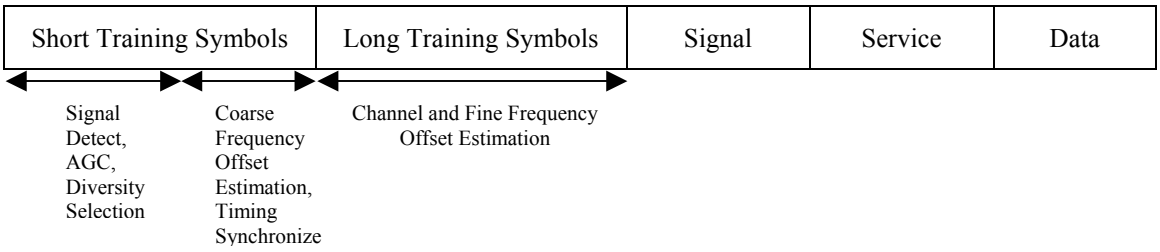

Figure 72 OFDM Training Structure [After Reference 10.]

The first six or seven short training symbols are intended to be used for the OFDM packet detection, setting the automatic gain control, and diversity antenna selection. The last three or four short training symbols are intended for coarse frequency estimation and initial timing synchronization. This coarse estimation is required by the receiver to perform small corrections without losing synchronization with the long training symbols. In addition to the long and short training symbols, the OFDM packet contains a pilot tone sequence interspersed with the data to maintain synchronization during the packet reception. [10]

There are many methods used to synchronize with OFDM signals, depending on the signal characteristics and the channel characteristics. Here we examine the optimum synchronization based on a channel model subject to additive white Gaussian noise (AWGN). The fundamentally different characteristics of an OFDM signal compared to a single-carrier signal make synchronization with OFDM signals a difficult problem.

A. PACKET DETECTION

The detection of an incoming packet in the presence of noise in the transmitting channel can be optimized by using binary hypothesis testing [16]. Hypothesis zero is when the input to the receiver is noise only. Hypothesis one is when the input to the receiver is a Standard 802.11g packet plus noise. It is assumed that the noise is AWGN. Mathematically this can be represented as

$$\begin{aligned} H_0 &: r(t) = n(t) \\ H_1 &: r(t) = s(t) + n(t) \end{aligned} \tag{4.1}$$

where H_0 is hypothesis zero, H_1 is hypothesis one, $r(t)$ is the received signal, $s(t)$ is the transmitted signal, and $n(t)$ is the noise.

The goal is to maximize the probability that the signal is detected in the presence of noise, and to minimize the probability that the receiver falsely detects noise only as a transmitted signal. Unfortunately both of these probabilities cannot be optimized simultaneously. If the decision threshold is changed to improve the probability of detecting a signal when present then the probability of false alarm also increases. The converse is also true. This can be seen graphically in Figure 73 using AWGN as the noise in the channel.

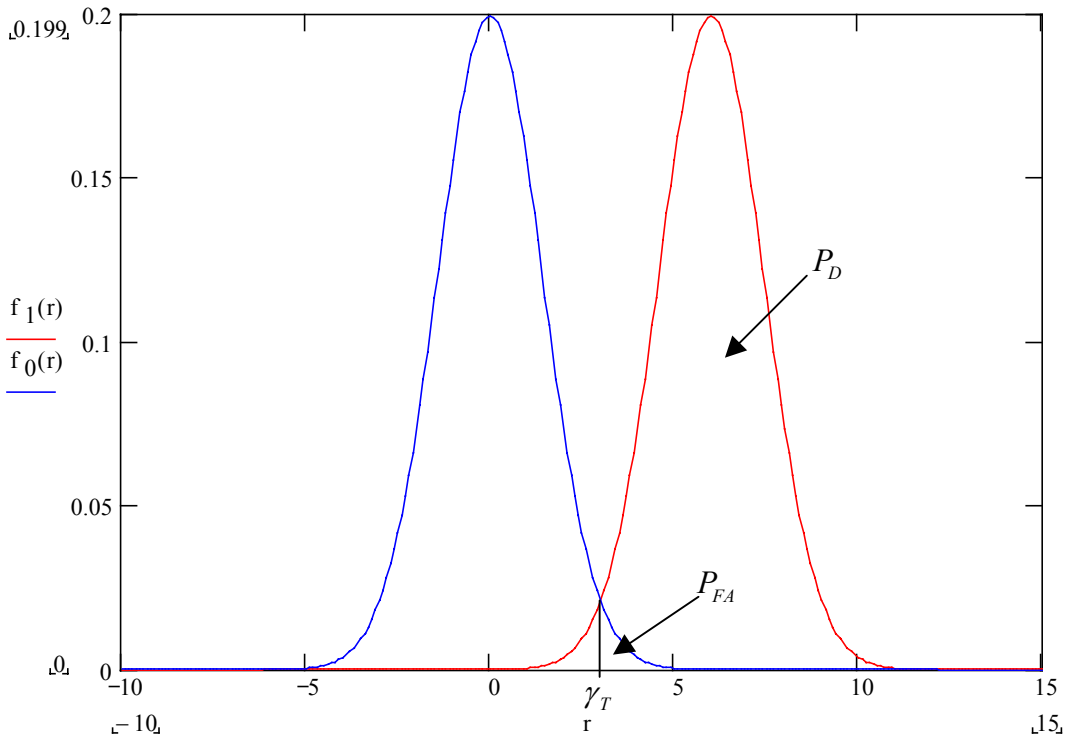

Figure 73 Probabilities of False Alarm and Detection in an AWGN Channel.

The curve on the left, $f_o(r)$, or the probability density function of the input $r(t)$ given hypothesis H_0, is shown as a Gaussian distributed, zero mean signal. It describes the input at the receiver under hypothesis H_0, when only the channel noise is present.

The curve on the right, $f_1(r)$, or the probability density function of the input $r(t)$ given hypothesis H_1, is shown as a Gaussian distributed signal with an arbitrary mean value of six which separates the two curves enough to demonstrate the tradeoffs made when selecting a receiver threshold. It represents the input at the receiver under hypothesis H_1, when the signal and the noise are present. The parameter $r(t)$ represents the decision statistic at the receiver, and the parameter γ_T represents the threshold between the receiver declaring that the receiver input represents noise only (hypothesis H_0) or signal and noise (hypothesis H_1). If the receiver detects a decision statistic below γ_T, it decides that no signal is present and does not start the detection and demodulation process. If the decision statistic exceeds γ_T, the receiver assumes that a signal is present at the input and commences with detection and demodulation. If the decision statistic exceeds γ_T because of the noise level, but no signal was transmitted, then a false alarm results. The area under the blue curve to the right of the threshold γ_T represents the probability of this occurring. If we increase the threshold γ_T, then the probability of false alarm is reduced as desired. However, the area under the red curve to the right of the threshold γ_T, representing the probability of signal detection when a signal is present, is also reduced. Increasing the threshold reduces the probability of false alarm, but also reduces the probability of detecting a signal when it is present. [16]

Likewise, the probability of detection can be increased at the expense of increasing the probability of a false alarm. This minimizes transmission interruptions due to the signal not being detected. The receiver can quickly recognize the false alarms if the anticipated training sequences are not received. The false alarms do not affect data transmission as long as the receiver can recognize a false alarm before the next packet is received. In Standard 802.11g compliant systems, the minimum spacing between packets, or the short interframe spacing interval (SIFS), is 16 μs. This is the length of the short and long training symbols. Therefore, if the receiver realizes that the signal it is trying to process is a false alarm before a time interval of 16 μs has passed, the chance of a packet

being transmitted during this interframe spacing will be minimized and packets missed due to the receiver being busy processing the false alarm will be reduced.

1. Optimum Packet Detection in AWGN

The optimum threshold is found by choosing a value such that hypothesis H_1 is chosen when the probability of hypothesis H_1 is greater than hypothesis H_0 given a decision statistic r_m [16]. Mathematically, this is expressed as

$$\Pr\left(H_1 \mid r_m\right) \underset{H_0}{\overset{H_1}{>}} \Pr\left(H_0 \mid r_m\right) \tag{4.2}$$

where r_m is the decision statistic at time $t = m$. Using the definition of probability and the probability distribution function of r_m under H_0 as $f_0\left(r_m\right)$ and the probability distribution function of r_m under H_1 as $f_1\left(r_m\right)$, Reference [16] shows that (4.2) can be represented as

$$\Lambda\left(r_m\right) = \frac{f_1\left(r_m\right)}{f_0\left(r_m\right)} \underset{H_0}{\overset{H_1}{>}} \lambda \tag{4.3}$$

where λ is the required threshold and $\Lambda\left(r_m\right)$ is the likelihood ratio.

The optimum value for the threshold of (4.3) depends on the criteria, or definition of optimum, used. Since in this case the concern is the detection of an incoming packet, the Neyman-Pearson criteria [16] is used. In the Neyman-Pearson criteria, the threshold λ is determined by the false alarm or detection probabilities. The probability of detection is defined as

$$P_D = \int_{\gamma_T}^{\infty} f_1(r_m)\, dr_m \tag{4.4}$$

where γ_T is the desired threshold.

Note that λ and γ_T are not necessarily equal. The decision is based on comparing the likelihood ratio to λ, or equivalently comparing r_m to γ_T.

If we assume that the noise is AWGN with a mean value of zero and a variance of σ_n^2 and channel fading is negligible, then $f_1(r_m)$ is Gaussian distributed with a mean value offset by the value of $s(t=m)=s_m$, and the variance remains unchanged. This means $f_1(r_m)$ is given by

$$f_1(r_m)=\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(r_m-s_m)^2}{2\sigma_n^2}} \tag{4.5}$$

where s_m is the value of $s\left(t=m\right)$. The probability density function under hypothesis H_0 is the same as that for the AWGN and is given by

$$f_0(r_m)=\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(r_m)^2}{2\sigma_n^2}}. \tag{4.6}$$

The likelihood ratio, or the ratio of the two density functions, is given by

$$\Lambda(r)=\frac{f_1(r)}{f_0(r)}=\frac{\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(r_m-s_m)^2}{2\sigma_n^2}}}{\frac{1}{\sqrt{2\pi\sigma_n^2}}e^{-\frac{(r_m)^2}{2\sigma_n^2}}}=e^{\frac{1}{2\sigma_n^2}\left[(r_m)^2-(r_m-s_m)^2\right]}. \tag{4.7}$$

Taking the log of both sides and simplifying, we get

$$\begin{aligned}\ln\left(\Lambda(r)\right)&=\frac{1}{2\sigma_n^2}\left[\left(r_m-r_m+s_m\right)\left(r_m+r_m-s_m\right)\right]\\&=\frac{1}{2\sigma_n^2}\left[s_m\left(2r_m-s_m\right)\right].\end{aligned} \tag{4.8}$$

Inserting (4.8) into inequality (4.3) and solving for r_m, we obtain

$$r_m \underset{H_0}{\overset{H_1}{\gtrless}} \frac{\sigma_n^2}{s_m} \ln \lambda + \frac{s_m}{2} = \gamma_T. \tag{4.9}$$

Inequality (4.9) specifically shows that the threshold chosen for the decision statistic r_m is not the same as λ. If we let $\gamma_T = \left(\sigma_n^2 / s_m\right) \ln \lambda + \left(s_m / 2\right),$ then (4.9) reduces to

$$r_m \underset{H_0}{\overset{H_1}{\gtrless}} \gamma_T. \tag{4.10}$$

Since the probability distribution of the received signal is known under both hypotheses, γ_T of (4.10) can be determined by specifying the desired probability of false alarm or the probability of detection. Using (4.4) to solve for the threshold $\gamma_T,$ we get

$$P_D = Q\left(\frac{\gamma_T - s_m}{\sigma_n}\right) \Rightarrow \gamma_T = s_m + \sigma_n Q^{-1}\left(P_D\right) \tag{4.11}$$

where $Q(x)$ is known as the Q-function and is defined as $Q(x) = \left(1/\sqrt{2\pi}\right) \int_x^\infty e^{-\frac{u^2}{2}}\, du$.

Note that the probability of false alarm is defined as

$$P_{FA} = \int_{\gamma_T}^{\infty} f_0(r_m)\, dr_m. \tag{4.12}$$

Combining (4.12) and (4.11), we have

$$P_{FA} = Q\left(\frac{\gamma_T}{\sigma_n}\right) = Q\left(\frac{s_m + \sigma_n Q^{-1}\left(P_D\right)}{\sigma_n}\right) = Q\left(\frac{s_m}{\sigma_n} + Q^{-1}\left(P_D\right)\right). \tag{4.13}$$

Assuming the AWGN is zero mean and variance of $\sigma_n^2 = 0.1$ and that the desired P_D is 0.9, we get a threshold of $\gamma_T = s_m - 0.40526 = 0.5947$ for $s_m = 1$. If the transmitted signal, s_m, has a value of one, this results in a $P_{FA} = 0.0300$.

In practice, a probability of detection greater than 0.9 must be used, since some errors will also occur when a packet presence is correctly detected, resulting in a loss of the packet.

The results of (4.10) can be implemented with a comparator circuit. Figure 74 shows a simple circuit set up in System View to demonstrate this packet detection technique.

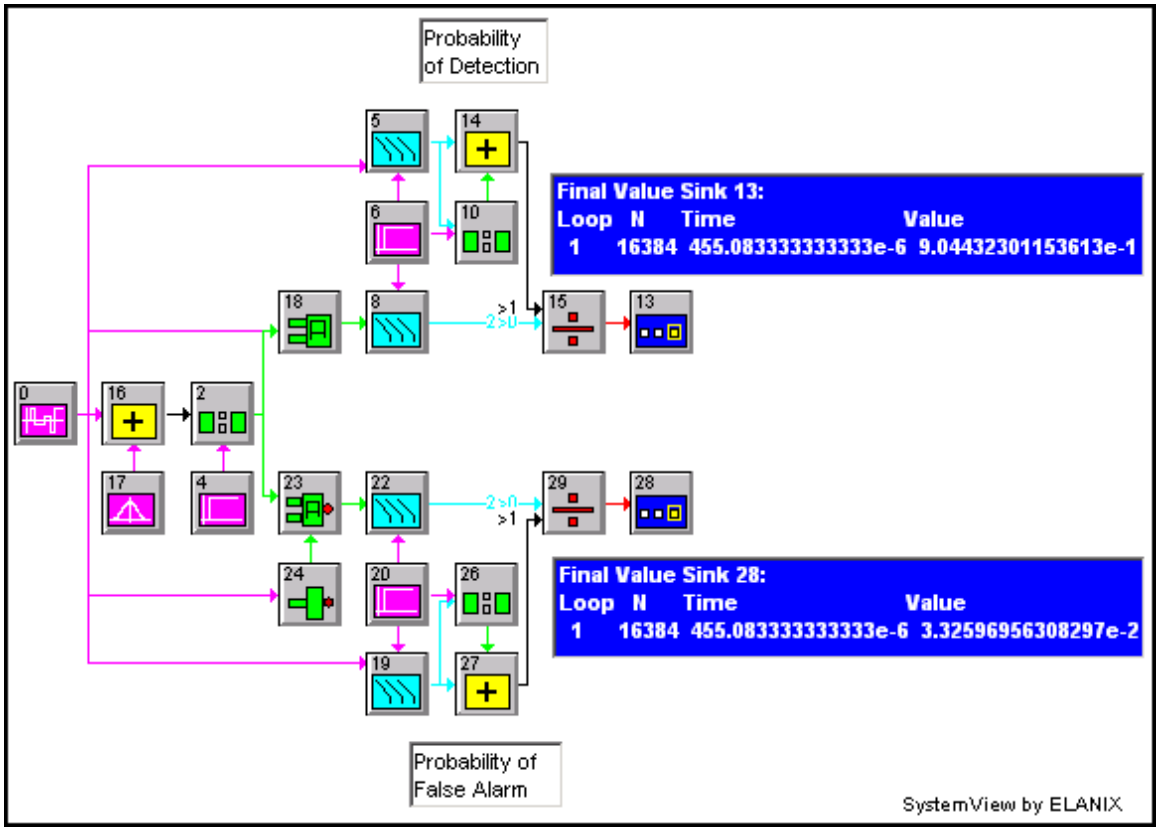

Figure 74 Probability Calculation Circuit.

The error rates shown in the final value sink windows are for a threshold value of $\gamma_T = 0.5947$ when the signal is $s_m = 1.0$. The simulated probability of detection is 90.4%, close to the value of 90% used in the analysis. The simulated probability of false alarm is 3.3%, matching the analysis earlier. When this system is run with a threshold value of $\gamma_T = 0.2$, the probability of false alarm is 27.3% and the probability of detection is

99.4%. This demonstrates the trade off between probability of detection and probability of false alarm.

2. Optimum Packet Detection Using Multiple Samples in AWGN

The detection capabilities of this circuit can be improved by separating the two probability density curves of Figure 73. Since changing the receiver design to decrease noise power is not an acceptable option due to cost or complexity, then the noise only density function (left) cannot be changed. However, the curve on the right can be shifted by proper design of the receiver and transmitter. The easiest way to separate the two curves is simply to increase the transmitted power. This increases the mean value (s_m) of the curve and separates the two probability densities. Simply increasing s_m from 1.0 to 2.0 in the circuit of Figure 74 and keeping the detection threshold at $\gamma_T = 0.5947$ keeps the probability of false alarm unchanged at 3.3%, but increases the probability of detection to 99.9%, a sizeable improvement over the 90.4% from the lower signal level.

Signal detection can also be improved by taking several samples of the input. This has the effect of shifting the signal plus noise density function (right hand side of Figure 73) to the right, but it also increases the variance, spreading both distributions out, as shown in Figure 75 where two samples are taken using the same parameters of Figure 73. This reduces the effectiveness of the curve shift in increasing the probability of detection or reducing the probability of false alarm.

For multiple sample values, the noise is uncorrelated because it is assumed white. This allows the resulting probability density function under each of the hypotheses to be factored into a product of the probability densities for each sample used in the receiver decision function. Reference [16] details the derivation of the resulting likelihood function when the noise is AWGN. Inequality (4.3) can then be modified to read

$$\Lambda(\mathbf{r}) = \frac{f_1(r_1) f_1(r_2) \cdots f_1(r_k)}{f_0(r_1) f_0(r_2) \cdots f_0(r_k)} \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \lambda. \tag{4.14}$$

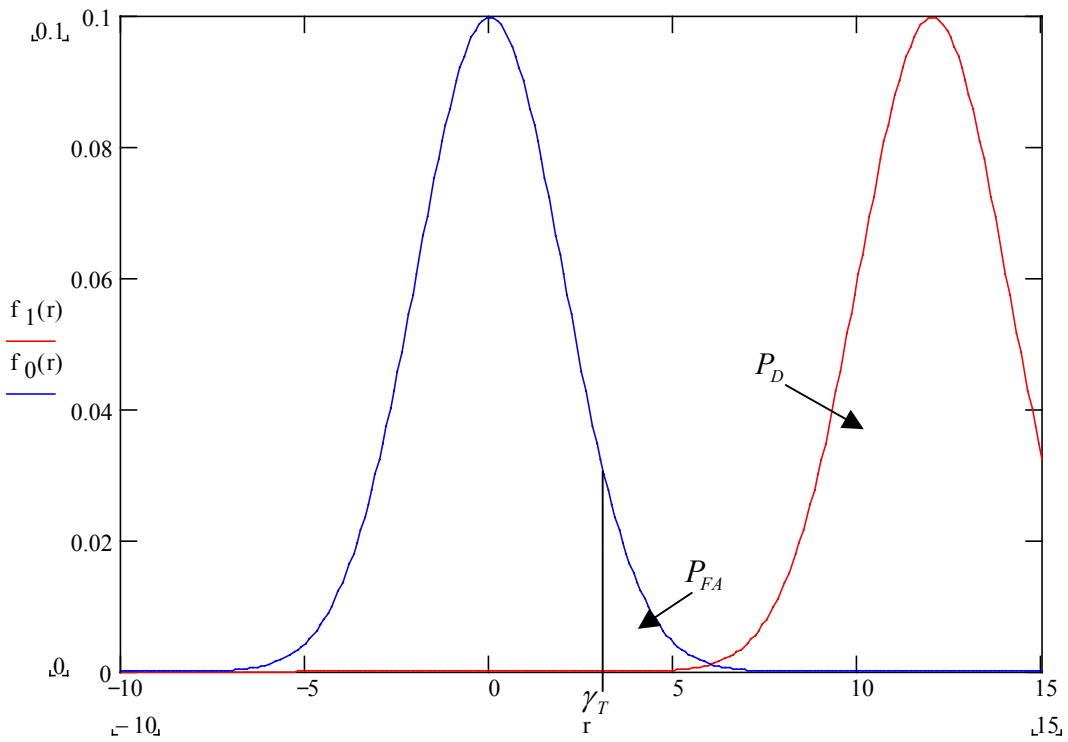

Figure 75 Change in Probability with Increased Samples.

The two hypotheses of (4.1) are rewritten to take into account the use of multiple signals:

$$\begin{aligned} H_0 &: r_m = n_m \\ H_1 &: r_m = s_m + n_m \end{aligned} \qquad 1 \le m \le k \tag{4.15}$$

where n_m is the AWGN sample at $t = m$. Substituting the Gaussian distributions consistent with (4.15) into (4.14), we get

$$\sum_{m=1}^{k} r_m s_m \underset{H_0}{\overset{H_1}{\gtrless}} \sigma_n^2 \ln(\lambda) + \frac{1}{2}\sum_{m=1}^{k}\left(s_m^2\right). \tag{4.16}$$

As before, the threshold on the right hand side of (4.16) can be consolidated into $\eta = \sigma_n^2 \ln(\lambda) + (1/2)\sum_{m=1}^{k}\left(s_m^2\right)$, resulting in

$$\sum_{m=1}^{k} r_m s_m \underset{H_0}{\overset{H_1}{\gtrless}} \eta. \tag{4.17}$$

The threshold is calculated as before using the definitions for probability of detection, P_D, and the probability of false alarm, P_{FA}, to calculate the threshold η. For the P_D, (4.4) becomes

$$P_D = \int_{\eta}^{\infty} f_{x1}(x)\, dx \tag{4.18}$$

where $x = \sum_{m=1}^{k} r_m s_m$ is the decision statistic and $f_{x1}(x) = \left(1/\sqrt{2\pi\sigma_x^2}\right) e^{-\frac{(x-m_x)^2}{2\sigma_x^2}}$ is the probability distribution under hypothesis H_1, since the sum of Gaussian random variables is Gaussian.

Using the same AWGN example as used in (4.11), we obtain

$$P_D = Q\left(\frac{\eta - m_{x1}}{\sigma_x}\right) \Rightarrow \eta = m_{x1} + \sigma_x Q^{-1}(P_D) \tag{4.19}$$

where m_x is the mean value of the decision statistic x under hypothesis H_1 and σ_x is the standard deviation of the decision statistic under hypothesis H_1. Taking the expected value of the decision statistic under hypothesis H_1, we get

$$m_{x1} = E\left\{\sum_{m=1}^{k} r_m s_m\right\} = E\left\{\sum_{m=1}^{k} (s_m + n_m) s_m\right\} = \sum_{m=1}^{k} \left(s_m^2 + s_m E\{n_m\}\right) = k s^2 \tag{4.20}$$

where the last step holds if $s_m = s$ is a constant. The variance of the decision statistic is

$$\begin{aligned}\sigma_x^2 &= E\left\{\left(\sum_{m=1}^{k}\left(s+n_m\right)s - m_{x1}\right)^2\right\} \\ &= E\left\{\left(\sum_{m=1}^{k}s^2 + \sum_{m=1}^{k}sn_m - \left(ks^2\right)\right)^2\right\} \\ &= E\left\{\left(\sum_{m=1}^{k}sn_m\right)^2\right\}\end{aligned} \tag{4.21}$$

$$\begin{aligned}\sigma_x^2 &= s^2 E\left\{\sum_{l=1}^{k}\sum_{m=1}^{k}n_l n_m\right\} \\ &= s^2\left(\sum_{l=1}^{k}\sum_{m=1}^{k}E\left\{n_l n_m\right\}\right) \\ &= s^2\left(\sum_{l=1}^{k}\sum_{m=1}^{k}\left[\left(E\left\{n_m^2\right\}\right)\delta_{l,m}\right]\right) \\ &= s^2\left[k\left(\sigma_n^2\right)\right] \\ &= k\sigma_n^2 s^2\end{aligned}$$

where $\delta_{l,m} = \begin{cases}1 & \text{if } l = m \\ 0 & \text{if } l \neq m\end{cases}$ is the Kronecker delta function.

Using (4.21) and (4.20) in (4.19), we find a threshold value of

$$\eta = ks^2 + \sqrt{k}s\sigma_n Q^{-1}\left(P_D\right). \tag{4.22}$$

Under hypothesis H_0, the mean value of the decision statistic is

$$m_{x0} = E\left\{\sum_{m=1}^{k}n_m s_m\right\} = 0. \tag{4.23}$$

The variance does not change from (4.21).

The probability of false alarm for the decision variable x is given by

$$P_{FA} = \int_{\eta}^{\infty} f_{x0}\left(x\right)dx \tag{4.24}$$

where $f_{x0}(x) = \left(1/\sqrt{2\pi\sigma_x^2}\right) e^{-\frac{x^2}{2\sigma_x^2}}$ is the probability distribution for x under hypothesis H_0. Using (4.22) and (4.23) in (4.24), we get the probability of false alarm as

$$\begin{aligned} P_{FA} &= Q\left(\frac{\eta}{\sigma_x}\right) \\ &= Q\left(\frac{ks^2 + \sqrt{k}s\sigma_n Q^{-1}(P_D)}{\sqrt{k}s\sigma_n}\right) \\ &= Q\left(\sqrt{k}\,\frac{s}{\sigma_n} + Q^{-1}(P_D)\right). \end{aligned} \tag{4.25}$$

Arbitrarily setting $s = \sigma_n = 1$ for illustrative purposes and using two sample values to demodulate the received signal ($k = 2$) results in a threshold of $\eta = 2 - 1.8124$ or $\eta = 0.1876$ when the probability of detection is 90%. Figure 76 shows a circuit that implements this detector using two samples of the generated random data sequence. The probability of detection for this simulation run is 89.5% when 8192 trials were used, close to the prescribed value of 90%. The simulated probability of false alarm calculated by the circuit of Figure 76 is 45.1%, which is better than the 61.7% we achieved when using only one sample of the received data sequence.

Equation (4.25) suggests that the probability of false alarm can be made arbitrarily small by increasing the number of samples used in the signal demodulation. For a constant P_D, the argument of the Q-function increases with increasing *k*, which causes P_{FA} to approach zero. Figure 77 shows this graphically.

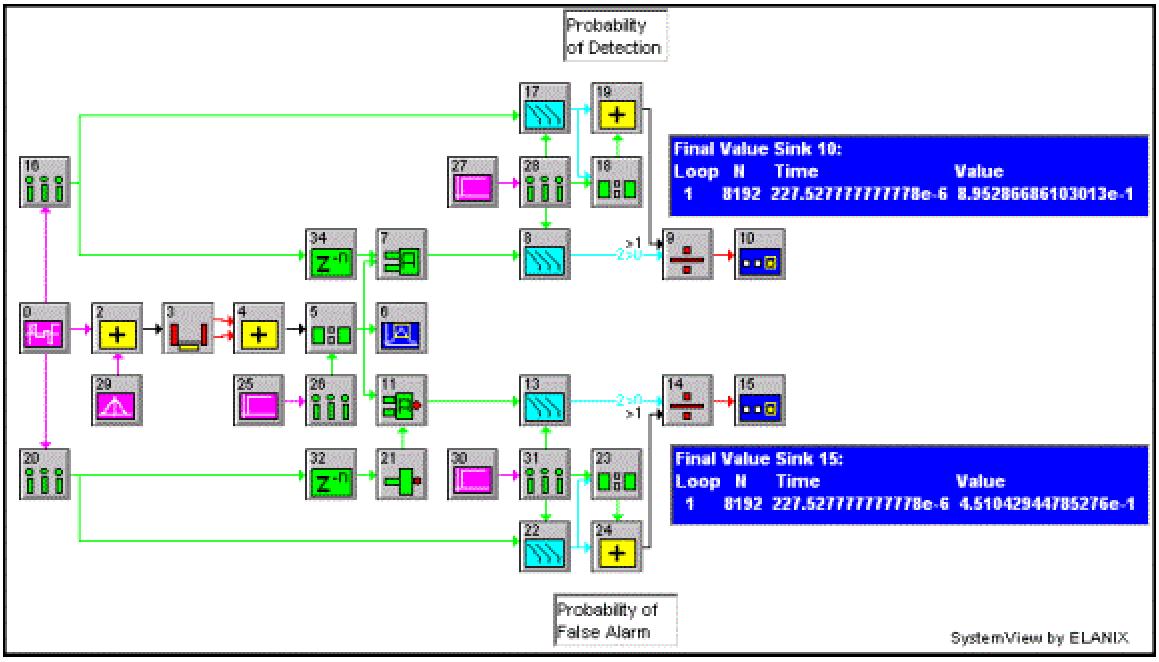

Figure 76 Multiple Sample Error Rate Calculation Circuit.

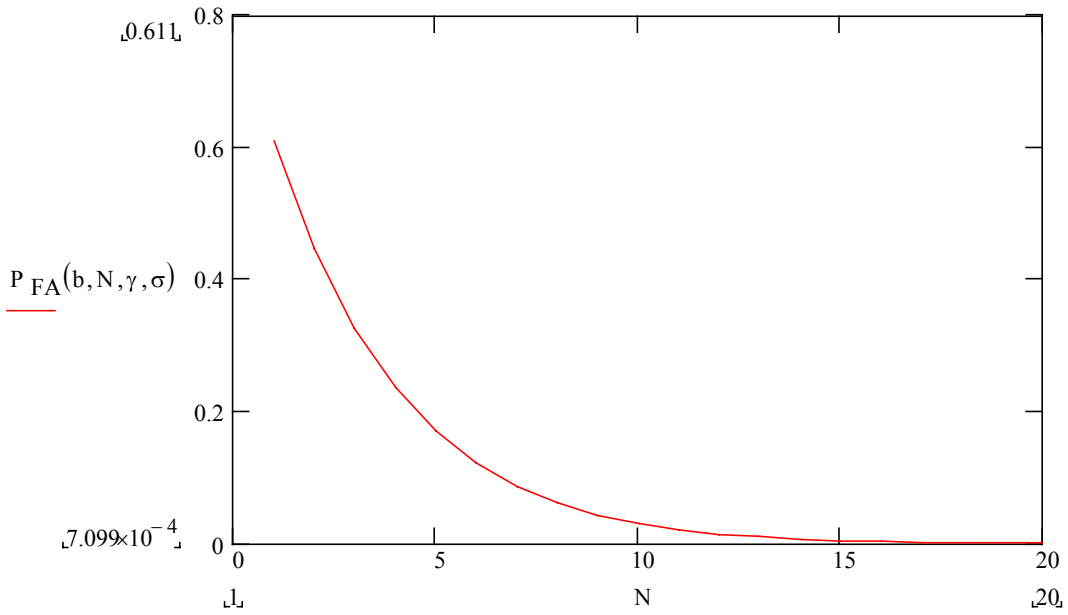

Figure 77 Probability of False Alarm vs. Sample Number.

In practice the number of samples that can be used to determine the existence of a signal is limited. This is especially true in packet-based systems such as the 802.11 stan-

dards. Reference [10] suggests using the first seven short training symbols for packet detection.

3. Other Packet Detection Designs

Two other packet-detection schemes mitigate the noise received to enhance the performance while maintaining the form of the optimum detection algorithms developed so far.

The first scheme uses multiple samples as detailed in Section 2 above, but uses two adjacent sliding windows to obtain the samples and then divides the output of the two windows to obtain the decision statistic [3]. Specifically,

$$\frac{\sum_{m=0}^{M-1}|r_m|^2}{\sum_{l=1}^{L}|r_{M+l}|^2} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{4.26}$$

where M is the length of the first sliding window and L is the length of the second (adjacent) sliding window. The advantage of this detector is that it gives a sharp pulse at the packet start time.

The second scheme uses a method similar to the first, but the incoming data is correlated with the known training data in the standard instead of just the sum of magnitudes squared. Since the short training symbols are periodic with a period of 16 samples in Standard 802.11g, a delayed version of the received signal is correlated with the received signal where the delay is 16 samples for Standard 802.11g signals. The structure of the receiver is shown in Figure 78.

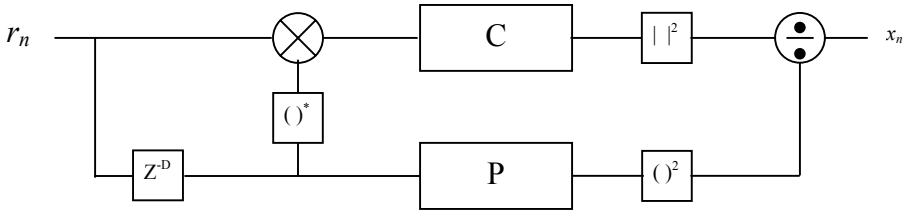

Figure 78 Delay and Correlate Detector [From Reference 3.]

In Figure 78, $C = \sum_{k=0}^{L-1} r_{n+k} r_{n+k-D}^{*}$ and $P = \sum_{k=0}^{L-1} r_{n+k-D} r_{n+k-D}^{*} = \sum_{k=0}^{L-1} \left| r_{n+k-D} \right|^2$. The output from this circuit is given by

$$x_n = \frac{|C|^2}{(P)^2} = \frac{\left| \sum_{k=0}^{L-1} r_{n+k} r_{n+k-D}^{*} \right|^2}{\left(\sum_{k=0}^{L-1} r_{n+k-D} r_{n+k-D}^{*} \right)^2} \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \eta. \tag{4.27}$$

The probability analysis of these two schemes is similar, since both deal with a ratio of sums of squares of Gaussian random variables. However, making a Gaussian approximation for x_n of (4.27) makes the analysis simpler [17]. Assuming the received signals are corrupted with Gaussian noise and that the variance of the noise is much smaller than the mean, we can approximate the probability distribution of x_n by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{4.28}$$

Schmidt and Cox [17] show that the mean value of the Gaussian approximation to x_n of (4.28) is given by

$$\mu = \frac{\sigma_s^4}{\left(\sigma_s^2 + \sigma_n^2 \right)^2} \tag{4.29}$$

where σ_n^2 is the variance of the AWGN. Schmidt and Cox [17] define $\sigma_s^2 = E\left\{ \mathrm{Re}\left[s_m \right]^2 \right\}$ and $\sigma_s^2 = E\left\{ \mathrm{Im}\left[s_m \right]^2 \right\}$, where the signal s_m is the value of the training data. The variance of (4.28) is given by

$$\sigma^2 = \frac{2\sigma_s^4 \left[(1+\mu)\sigma_s^2 \sigma_n^2 + (1+2\mu)\sigma_n^4 \right]}{16\left(\sigma_s^2 + \sigma_n^2 \right)^4}. \tag{4.30}$$

The threshold, probability of detection, and probability of false alarm are calculated using (4.29) and (4.30) in the same manner as in Sections one and two.

B. FREQUENCY SYNCHRONIZATION

Frequency error sensitivity is a weakness of OFDM systems, since small changes in the sub-carrier frequency caused by distortions in either the channel or the receiver can make the sub-carriers loose their orthogonality. Once this occurs, the interference between adjacent sub-carriers becomes significant and the received signal level is reduced. The complex envelope of the transmitted Standard 802.11g OFDM symbol has the form of [10]

$$r_n(t) = w_n(t) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_{k,n} e^{j2\pi k\Delta f(t-T_{GUARD})} \tag{4.31}$$

where $w_n(t)$ is the windowing function for the n^{th} OFDM symbol, $C_{k,n}$ is the coded symbol modulating the k^{th} sub-carrier in the n^{th} OFDM symbol, N_{ST} is the total number of sub-carriers, Δf is the sub-carrier frequency spacing, and T_{GUARD} is the guard interval that created the circular prefix used to avoid inter-symbol interference (ISI) and inter-carrier interference(ICI).

In order for the sub-carriers to be orthogonal, the following relationships must apply [1]:

$$\int_{-\infty}^{\infty} r_n(t) r_m^*(t)\, dt = 0 \quad n \neq m \tag{4.32}$$

$$\int_{-\infty}^{\infty} w_n(t)\left(C_{k,n} e^{j2\pi k\Delta f(t-T_{GUARD})}\right)\left(C_{l,n} e^{j2\pi l\Delta f(t-T_{GUARD})}\right)^* dt = 0 \text{ for } k \neq l. \tag{4.33}$$

Equation (4.32) ensures that adjacent OFDM symbols are orthogonal, and (4.33) ensures the sub-carriers in each OFDM symbol are orthogonal. This mitigates ISI in the time-domain and ICI in the frequency-domain. Defining the windowing function in (4.31) and (4.33) as $w_n(t) = \begin{cases} 1 \text{ for } T_{GUARD} \le t \le T \\ 0 \text{ for all other } t \end{cases}$, we get a simplification of (4.33) as

$$\int_{T_{GUARD}}^{T} \left(C_{k,n} e^{j2\pi k\Delta f(t-T_{GUARD})}\right)\left(C_{l,n}^* e^{-j2\pi l\Delta f(t-T_{GUARD})}\right) dt = 0 \text{ for } k \neq l. \tag{4.34}$$

Equation (4.34) is only satisfied when

$$e^{j2\pi(k-l)\Delta f(T-T_{GUARD})}-1=0 \text{ or } 2\pi(k-l)\Delta f(T-T_{GUARD})=2\pi m \text{ for some } m=\pm1,\pm2,\ldots \quad (4.35)$$

Equation (4.35) is true for all $k \neq l$ if and only if there exists a positive integer m such that

$$\Delta f = \frac{m}{T_{FFT}} \quad (4.36)$$

where $T_{FFT} = T - T_{GUARD} = 3.2\ \mu s$ for Standard 802.11g. The minimum spacing between sub-carriers occurs when $m = 1$, and (4.36) simplifies to

$$\Delta f = \frac{1}{T_{FFT}} = 0.3125 \text{ MHz}. \quad (4.37)$$

Equation (4.37) gives the required frequency spacing between sub-carriers for the transmitter of a Standard 802.11g compliant system. The required bandwidth for a Standard 802.11g system is

$$B_T = (52-1)(0.3125 \text{ MHz}) + BW_{data}. \quad (4.38)$$

This accounts for the bandwidth of the signals contained within the 52 sub-carriers, 48 of which carry data, of each OFDM symbol. The data bandwidth is the same for all data rates because of the change in coding rates. At the 6-Mbps data rate using BPSK (or equivalently 12 Mbps using QPSK, or 24 Mbps using 16-QAM, etc.), the null-to-null data bandwidth is twice the data rate in one channel of transmitted data. Since the 6-Mbps data is coded using a rate $r = 1/2$ encoder (with no puncturing), the coded data rate is 12 Mbps. Dividing this data signal between 48 sub-carriers, we obtain a data bandwidth of $BW_{data} = 2\left[\left(12\times10^6\right)/48\right] = 0.5$ MHz. Using this value of null-to-null bandwidth in (4.38), we get a total null-to-null bandwidth of $B_T = 16.4375$ MHz, which is less than the maximum bandwidth of 16.6 MHz specified in the standard. The 16.6 MHz must include most of the power in the spectral tails that extend beyond the null-to-null bandwidth.

Frequency recovery schemes for OFDM signals can be divided into three categories: non-data aided algorithms that are based on the spectral characteristics of the received signal, cyclic prefix based algorithms that use the structure of the signal in an

OFDM system provided by the cyclic prefix, described in Chapter I Section 2, and data-aided algorithms that are based on known information embedded in the received signal [18]. Most frequency synchronization systems are similar to the phase-locked loop, as shown in Figure 79. The difficulty the OFDM systems present is that the multiple sub-carriers present in the received signal make it difficult to generate a single discrimination signal at the output of the multiplier that can be filtered and used as an error signal. Most of the algorithms used in optimum frequency synchronization generate an error signal proportional to the frequency offset that is used to drive a voltage-controlled oscillator (VCO). For OFDM frequency synchronization, the loop filter ($H(f)$) is much more complicated than the simple loop filter that can normally be used in a single-carrier phase-locked loop.

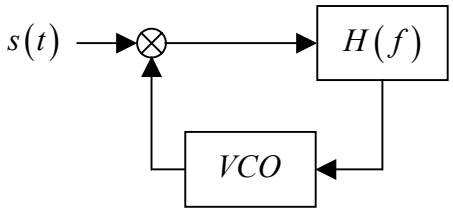

Figure 79 Phase-Locked Loop Configuration.

1. Non-Data Aided Frequency Synchronizers

The non-data aided synchronizers can be classified as open loop and closed loop. In an open loop synchronizer, a non-linear element, such as a squaring circuit, is used to generate a frequency component at a harmonic of the carrier frequency. The signal is then filtered to isolate this harmonic and stepped down to the desired carrier frequency. This type of synchronizer is generally used with signals in which the carrier has been suppressed and contains no spectral line at the carrier frequency, such as *M*-ary phase-shift keying (*M*-PSK). An unavoidable tracking error is always present in open loop systems. The advantage of these systems is their simplicity and low cost of implementation. However, because of the sensitivity of OFDM signals to frequency offset, open loop synchronizers are generally not practical for OFDM receivers.

Closed-loop synchronization uses comparative measurements on the incoming signal and a locally generated signal to bring the locally generated signal into synchronization with the incoming signal. Examples include phase-locked loops and delay-locked loops. For OFDM synchronizers that use closed-loop, non-data aided techniques, the basic process is to obtain the probability density function of the received signal for a given frequency offset, $p_r\left(r \mid \Delta f\right)$. Once this is obtained, the maximum-likelihood of the frequency offset for a given received signal is calculated by finding the value of Δf for which the conditional probability density function, $p_r\left(r \mid \Delta f\right)$, is a maximum. This frequency error signal is then fed into the voltage-controlled oscillator to adjust the estimated frequency. The analysis of the receiver is quite complex, and the acquisition range of these detectors is usually limited to $\pm 1/2$ of the sub-carrier frequency spacing due to the local maximum and minimum in the conditional probability density function, $p_r\left(r \mid \Delta f\right)$. The many variations of this technique attempt to reduce the calculations required and increase the acquisition range while still maintaining optimum or near optimum estimates.

Reference [19] presents an example of an algorithm that uses this maximum-likelihood technique. The block diagram of this synchronizer is shown in Figure 80. All signals in the diagram and accompanying analysis are complex envelope (i.e., equivalent lowpass) signals. Daffara and Chouly [19] derive the error signal at the output of the correlator, which is given by:

$$\varepsilon_n = \sum_{k=L_1}^{L_2} \mathrm{Re}\left\{q_{n,k}\dot{q}_{n,k}^{*}\right\} \tag{4.39}$$

where n is the time index, $L_1 = \left[\left(N - N_u\right)/2\right] + 1$ is the lowest index of the OFDM sub-carriers with N equal to the total number of sub-carriers used in the OFDM symbol and N_u equal to the number of non-zero sub-carriers, $L_2 = \left(N + N_u\right)/2$ is the highest index of the OFDM sub-carriers, $\mathrm{Re}\left\{x\right\}$ is the real part of x, $q_{n,k} = \int_T r(t) e^{-j2\pi f_k t} e^{-j\left(2\pi \tilde{f}_r t + \tilde{\phi}_r\right)}\, dt$ with $\tilde{f}_r$ and $\tilde{\phi}_r$ representing the received frequency and phase errors, and

$\dot{q}_{n,k} = -j2\pi \int_T r(t)\,\mathrm{tr}\left(\frac{t-nT}{T}\right) e^{-j2\pi f_k t} e^{-j\left(2\pi \tilde{f}_r t + \tilde{\phi}_r\right)}\, dt$ where T is the OFDM symbol duration. The triangle function $\mathrm{tr}(x)$ is defined as $\mathrm{tr}(x) = \begin{cases} 1-|x| & |x|<1 \\ 0 & |x|>1 \end{cases}$.

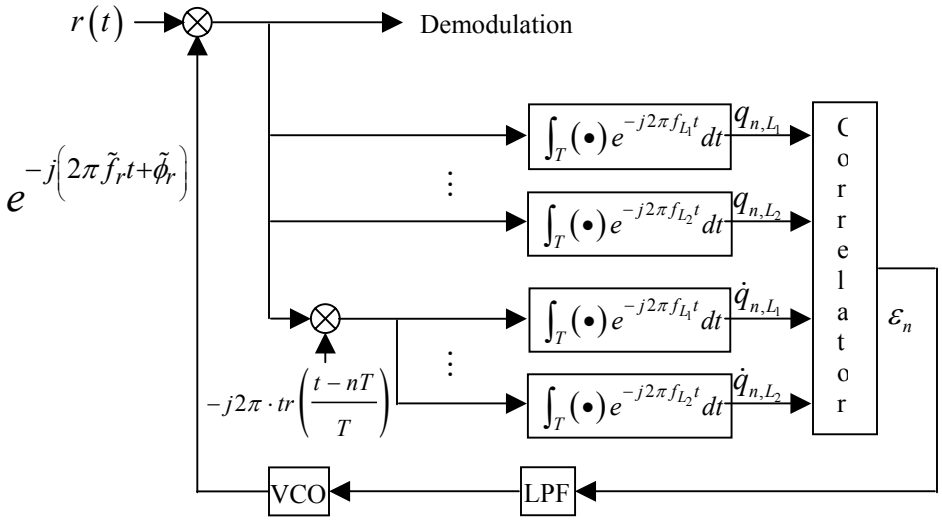

Figure 80 Maximum-Likelihood Frequency Detector [After Reference 19.]

Figure 80 shows how the frequency error signal is obtained. The correlator block computes the summation over *k* of $\mathrm{Re}\left\{q_{n,k}\dot{q}^*_{n,k}\right\}$ given by (4.39).

This example serves to illustrate the complexity of an optimum detector using non-data aided techniques. The author does offer some simplifications by using FFTs and smoothing the likelihood function, again with a loss of optimization. Daffara and Chouly show practical implementations of their algorithm and the characteristic curves for their designs in Reference [19].

Luise and Reggiannini in Reference [20] use a somewhat different approach to generate the error signal that drives a numerically controlled oscillator (NCO). The received signal is first demodulated with an FFT and an offset between the carrier frequency and the local oscillator frequency is estimated using a differential detector. This signal is corrected by the estimated frequency offset, modulated using an IFFT and com-

pared to the received signal to obtain a residual frequency error estimate. This residual frequency error estimate then drives a numerically controlled oscillator.

Non-data aided algorithms do not need special synchronization blocks, increasing the data throughput and reducing the time needed to achieve synchronization by eliminating the wait for synchronization blocks to appear in the data. For these reasons, these algorithms are well suited for continuous broadcast OFDM signals. However, performance is poor in the presence of multipath interference, frequently encountered in mobile radio environments. Packet-based OFDM signals are also not well suited to this kind of synchronization, since the accuracy is not great enough to ensure orthogonality during the entire packet transmission time.

2. Cyclic Prefix Based Frequency Synchronization

Cyclic prefix algorithms are based on an analysis of the sampled received signal before it is passed through the FFT for demodulation. They make use of the redundancy introduced by the inserted guard interval in the OFDM symbol. Since the guard interval is a repetition of the transmitted OFDM symbol over some fraction of the OFDM symbol period, these algorithms simply compare the samples from the data portion of the symbol and the corresponding samples from the guard interval portion of the symbol. If there is a frequency offset of Δf in the receiver frequency, the two values will be different by a factor of $e^{j2\pi\Delta fT}$ where T is the time difference between the two values. This phase difference is proportional to the magnitude of the frequency-offset error and can then be used as the error signal that drives a voltage-controlled oscillator. The computational complexities of these algorithms are less than the other two categories, therefore providing faster synchronization with lower hardware cost.

Reference [21] is an example of this type of algorithm. The data samples of the received OFDM symbol are designated as shown in Figure 81 where N is the number of data samples and N_g is the number of samples used in the guard interval. The guard interval is the cyclic extension of the N-value IFFT. This means that $c_{-1} = c_{N-1}$, $c_{-2} = c_{N-2}$, etc. In general $c_{-i} = c_{N-i}$ when the signal leaves the transmitter. If no frequency offset exists

between the transmitted and the received signal, then a sample in the guard interval (c_{-i}) multiplied by the conjugate of its corresponding sample value within the *N*-value IFFT (c_{N-i}) is a real number ($c_{-i} \cdot c_{N-i}^{*} = c_{N-i} \cdot c_{N-i}^{*} = \left|c_{N-i}\right|^2$). This also assumes that the received signal inside the guard interval has not been corrupted by intersymbol interference. If a frequency offset of Δf is introduced in the channel, then the product between the two received samples is $r_{-i} r_{N-i}^{*} = c_{-i} e^{j2\pi\Delta f t} \cdot c_{N-i}^{*} e^{-j2\pi\Delta f (t+T)} = \left|c_{N-i}\right|^2 \times \ e^{-j2\pi\Delta f T}$. This means the product $r_{-i} \cdot r_{N-i}^{*}$ is no longer real and it contains a phase offset proportional to Δf. Daffara and Adami [21] derive an error signal in the presence of AWGN given by

$$\varepsilon(l) = \tfrac{1}{L} \sum_{i=0}^{L} \mathrm{Im}\left\{r_{N-i} r_{-i}^{*}\right\} \tag{4.40}$$

where $1 \le L \le N_g$ is the number of guard interval samples used in the calculation and r_{N-i} and r_{-i} are the received samples. Using (4.40), we obtain the error signal $\varepsilon(l)$ proportional to the sine of the scaled frequency offset $2\pi\Delta f T$.

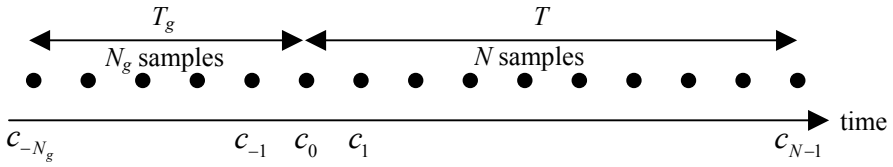

Figure 81 Sample Order in an OFDM Symbol [From Reference 21.]

The number of samples that can be used to generate the error signal is limited to those that have not been corrupted by intersymbol interference. One potential problem with this detector is its inability to compensate for frame synchronization error. The algorithm depends on knowing where the guard interval stops and the normal data sequence begins. This information may not be very accurate at the receiver. In addition, using too few samples of the guard interval can drastically affect the output. The frequency offset of this algorithm is limited $\pm 1/2$ of the sub-carrier spacing. The block diagram implementation of this algorithm is shown in Figure 82, where the input to the frequency detec-

tor can be either before the FFT (designated method A and described in the paragraph above) or after the FFT (designated as method B).

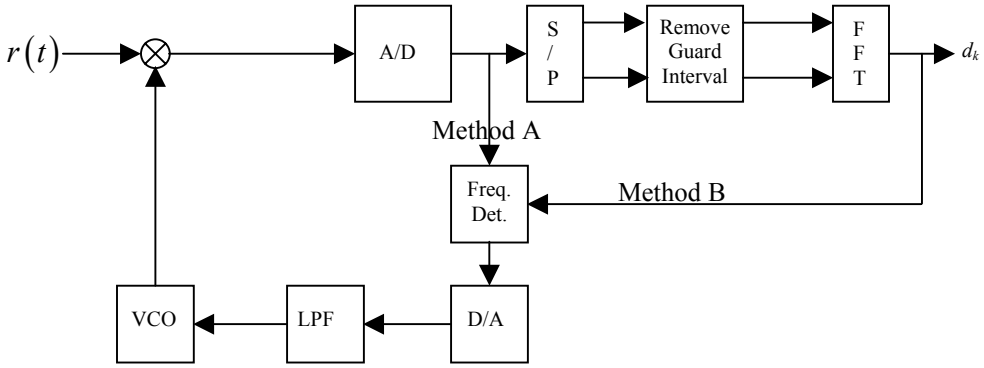

Figure 82 Guard Interval Based Frequency Detector [After Reference 21.]

Some of these concerns are addressed in the algorithms of Reference [22] and Reference [23]. Both of these algorithms incorporate frame timing to help estimate when the guard interval ends and when the normal sequence starts.

These algorithms are also more suited to a continuously transmitting OFDM signals. Since it takes several guard intervals to obtain synchronization, several OFDM symbols need to be transmitted, something that is not available in a Standard 802.11g packet-based system before synchronization must take place. However, this does eliminate the need for any special synchronization symbols, increasing the data throughput.

3. Data-Aided Frequency Synchronizers

Data-aided frequency synchronization provides the best frequency tracking with the widest acquisition range, but at the cost of requiring the use of synchronization blocks. This increases the required overhead and reduces the data throughput. However, for packet-based transmission systems, such as Standard 802.11g, they are required to obtain synchronization quickly before the data information is passed to the receiver. For Standard 802.11g systems, synchronization must occur within the short and long training symbols, which make up the first 16 μs of the packet. The basic algorithm assumes a se-

quence of repeated training symbols. Similar to the method used in the cyclic prefix algorithms, a comparison is made of the phase difference between adjacent, repeated data symbols. This phase difference is used to generate an error signal that drives a voltage-controlled oscillator.

Reference [3] details one example of a data-aided algorithm. Figure 83 shows a block diagram of this receiver. This technique compares the phase between demodulated values of successive repeated data symbols, making the assumption that the channel impulse response is stationary during the 8 μs OFDM symbol transmission interval of the training sequence. Since the training symbols do not change during the acquisition sequence, any difference in phase must be due to a frequency offset in the receiver.

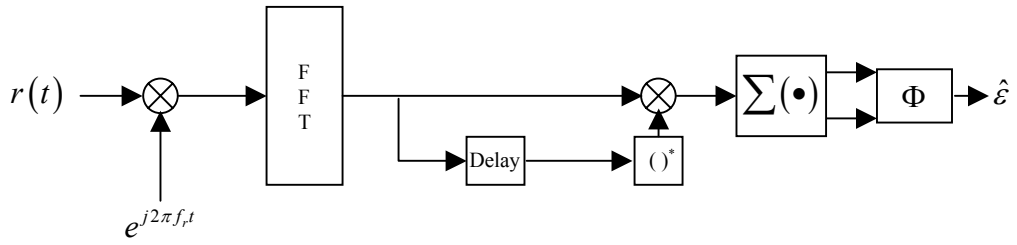

Figure 83 Frequency-Domain Data-Aided Frequency Circuit.

The equation for the n^{th} transmitted Standard 802.11g OFDM training symbol is given by (4.31). This signal is modulated with the carrier frequency to produce

$$r_n(mT_s) = w_n(mT_s)\left(\frac{1}{N_{ST}+1}\right)\sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta fnT_d}\cdot e^{j2\pi f_t mT_s} \tag{4.41}$$

where f_t is the carrier frequency of the transmitter, T_d is the data symbol period, T_s is the sample period, and m is the sample index ($t = mT_s$). The other terms have been defined in (4.31). Rearranging (4.41) and assuming the channel is noise free, we obtain the received signal

$$r_n(mT_s) = \left(\frac{1}{(N_{ST}+1)}\right)e^{j2\pi f_t mT_s}\sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta fnT_d}. \tag{4.42}$$

The window function $w(mT_s)$ has been dropped, since only one received OFDM symbol is being considered (i.e., $n(N_{ST}+1)T_d \le t \le (n+1)(N_{ST}+1)T_d$) and $w(mT_s)=1$ in this interval. The signal of (4.42) is demodulated with the receiver's estimate of the carrier frequency (f_r) to produce

$$x_n = \left(\frac{1}{(N_{ST}+1)}\right) e^{j2\pi(f_t-f_r)mT_s} \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f nT_d} \tag{4.43}$$

where f_r is the receiver estimate of the carrier frequency. Making the substitution of $f_\Delta = f_t - f_r$ into (4.43), we get

$$x_n = \left(\frac{1}{(N_{ST}+1)}\right) e^{j2\pi f_\Delta mT_s} \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f nT_d}. \tag{4.44}$$

If the training symbols C_k are repeated every OFDM symbol, then the received signal for the n^{th} data sample in the adjacent OFDM symbol is given by

$$\begin{aligned} x_{n+N} &= e^{j2\pi f_\Delta (n+N)T_d} \left(\frac{1}{(N_{ST}+1)}\right) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f (n+N)T_d} \\ &= e^{j2\pi f_\Delta NT_d} \left\{ e^{j2\pi f_\Delta nT_d} \left(\frac{1}{(N_{ST}+1)}\right) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f nT_d} e^{j2\pi k\Delta f NT_d} \right\}. \end{aligned} \tag{4.45}$$

If the receiver is designed such that the sample period, T_s, is an integer multiple of the data symbol period, T_d, then this enables the simplification of $t = mT_s = nT_d$ to be substituted in the phase error term. But the definition of orthogonality requires $\Delta f = 1/(NT_d)$ and $e^{j2\pi k\Delta f NT_d} = e^{j2\pi k[1/(NT_d)]NT_d} = e^{j2\pi k}$ and $e^{j2\pi k} = 1$. This allows a simplification of (4.45) to

$$x_{n+N} = e^{j2\pi f_\Delta NT_d} \left\{ e^{j2\pi f_\Delta nT_d} \left(\frac{1}{(N_{ST}+1)}\right) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f nT_d} \right\}. \tag{4.46}$$

Comparing (4.46) and (4.44), we get

$$x_{n+N} = e^{j2\pi f_\Delta N T_d} x_n. \tag{4.47}$$

Equation (4.47) shows that the data in adjacent OFDM symbols are offset in phase from each other equal to the phase error. Since the FFT is a linear process ($y_k = \sum_{k=-N_{ST}/2}^{N_{ST}/2} x_n e^{-j2\pi k \Delta f n T_d}$), the demodulated data symbols will also be offset in phase from each other and $y_{k+N} = e^{-j2\pi f_\Delta N T_d} y_k$. The contribution to the phase of the data symbols can be removed by complex multiplication $y_k \cdot y_{k+N}^* = e^{j2\pi f_\Delta N T_d} |y_k|^2$. Taking a summation over all the data symbols in one OFDM symbol, $z_k = \sum_{k=0}^{N-1} y_k \cdot y_k^*$, we average out the effect of any noise in the system. Since the only contribution to the phase of z_k is due to the frequency offset, the phase of z_k is used to produce the estimated frequency offset to the voltage-controlled oscillator. Reference [3] gives the final frequency estimate as

$$\hat{\varepsilon} = \left(\frac{1}{2\pi}\right) \tan^{-1} \left\{ \frac{\sum_{k=-K}^{K} \mathrm{Im}\left[Y_{2k} Y_{1k}^*\right]}{\sum_{k=-K}^{K} \mathrm{Re}\left[Y_{2k} Y_{1k}^*\right]} \right\} \tag{4.48}$$

where $\hat{\varepsilon} = f_\Delta N T_d$, Y_{2k} is the demodulated data symbol from the second OFDM symbol, and Y_{1k} is the demodulated data symbol from the first OFDM symbol. Reference [3] also shows that this algorithm is a maximum-likelihood algorithm in the presence of AWGN.

As can be seen from (4.48), the acquisition is limited to $-1/4 \le \hat{\varepsilon} \le 1/4$ or equivalently $-1/(4NT_d) \le f_\Delta \le 1/(4NT_d)$, which is a span of $1/2$ the sub-carrier spacing.

Reference [3] applies the same algorithm to the received data before it is demodulated with an FFT. Figure 84 shows a block diagram of this receiver.

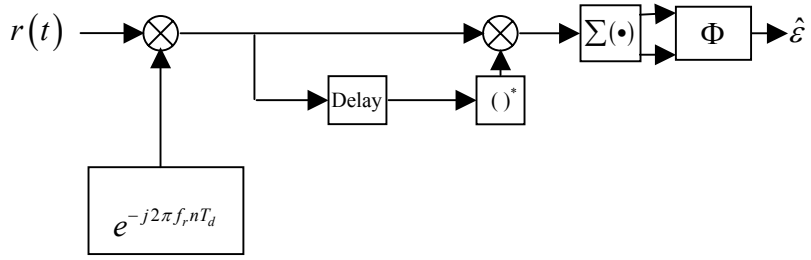

Figure 84 Time-Domain Data-Aided Frequency Circuit.

In this algorithm, the data symbols in the frequency-domain are repeated every l data symbols. This means

$$\begin{aligned} r_n &= \left(\frac{1}{(N_{ST}+1)}\right) e^{j2\pi f_t m T_s} \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f n T_d} \\ &= r_{n+l} \\ &= \left(\frac{1}{(N_{ST}+1)}\right) e^{j2\pi f_t (n+l) T_d} \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f (n+l) T_d}. \end{aligned} \tag{4.49}$$

Demodulation of (4.49) with the carrier frequency and simplification results in the same results as (4.44). For the repeated data symbol at an offset of l from the demodulated x_n, we obtain

$$x_{n+l} = e^{j2\pi f_\Delta (n+l) T_d} \left(\frac{1}{(N_{ST}+1)}\right) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f (n+l) T_d}. \tag{4.50}$$

Equation (4.50) can be rearranged to obtain

$$x_{n+l} = e^{j2\pi f_\Delta l T_d} \left\{ e^{j2\pi f_\Delta n T_d} \left(\frac{1}{(N_{ST}+1)}\right) \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f (n+l) T_d} \right\}. \tag{4.51}$$

However, since the received data symbols repeat every l samples, then

$$\sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f (n+l) T_d} = \sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f n T_d} \tag{4.52}$$

which allows (4.51) and (4.52) to be combined into

$$x_{n+l} = e^{j2\pi f_\Delta l T_d}\left\{e^{j2\pi f_\Delta n T_d}\left(\frac{1}{(N_{ST}+1)}\right)\sum_{k=-N_{ST}/2}^{N_{ST}/2} C_k e^{j2\pi k\Delta f n T_d}\right\} \qquad (4.53)$$
$$= e^{j2\pi f_\Delta l T_d} x_n$$

where x_n is taken from (4.44).

As was done in the frequency-domain case, the dependence of the data on the phase of x_n is removed by multiplying by the complex conjugate, and the noise is averaged out by summing all the data points between the repeated data symbols. The resulting frequency offset estimation is

$$\hat{\varepsilon} = -\frac{1}{2\pi}\tan^{-1}\left\{\frac{\sum_{n=0}^{L-1}\mathrm{Im}\left[x_n x_{n+l}^*\right]}{\sum_{n=0}^{L-1}\mathrm{Re}\left[x_n x_{n+l}^*\right]}\right\} \qquad (4.54)$$

where $\hat{\varepsilon} \approx f_\Delta l T_d$.

This same algorithm can be applied to both the long and short training symbols. For Standard 802.11g systems, the maximum frequency offset that can be corrected using the short training symbols is $f_{\Delta\max} = 1/(2lT_d) = 1/\left[2(16)\left(50\times10^{-9}s\right)\right] = 625$ kHz. The maximum frequency offset that can be corrected using the long training symbols is $f_{\Delta\max} = 1/(2lT_d) = 1/\left[2(64)\left(50\times10^{-9}s\right)\right] = 156.25$ kHz. The maximum allowed frequency error for a Standard 802.11a system is ± 20 parts per million. With a transmitter frequency of 5.3 GHz, this equates to a frequency error of 212 kHz. This is within the acquisition range of the short training symbols, but outside the acquisition range for the long training symbols. For Standard 802.11g, the maximum allowed frequency error is ± 25 parts per million. With a transmitter frequency of 2.4 GHz, this equates to a frequency error of 120 kHz, well within the tolerance of the short and long training symbol accuracy. The longer time between the long training symbols gives its frequency tracking more accuracy, but with less tolerance for initial offset error. The short training symbols provide the improved accuracy required by the long training symbols.

Both the frequency-domain estimator and the time-domain frequency estimator have the same performance, however, the time-domain algorithm does not require an FFT calculation, saving time in the frequency estimation step. Also, the larger frequency interval between the repeated data symbols in the frequency-domain estimator ($N > l$) results in a smaller tolerance for input frequency error.

A different approach to the data aided technique of synchronization can be found in Reference [24]. This technique uses virtual sub-carriers to estimate frequency offset and symbol timing. The virtual sub-carriers are those sub-carriers that contain no data and no signal power. This applies to Standard 802.11g, since only 52 of the 64 sub-carriers are used to transmit data or pilot signals. However, this algorithm is designed around a system that has multiple users (vice one user for a Standard 802.11g transmitter), with each user allocated some predetermined number of sub-carriers to use for data transmission. In either case, some sub-carriers are used for data and pilot tones and the rest are set to zero. If, after processing the signal through the FFT, any of the virtual sub-carriers contain a signal, then there is a frequency offset in the receiver. This algorithm adjusts the receiver frequency until the measured signal in the virtual sub-carriers is a minimum. Figure 85 shows a block diagram of this system. The advantage of this approach compared to the other data-aided algorithms is that it does not require synchronization symbols to be transmitted, similar to the guard interval synchronization methods mentioned earlier.

To find the value of frequency offset that minimizes the signal energy within the virtual sub-carriers, a cost function is defined that represents the average energy within all the virtual sub-carriers over several OFDM symbols as

$$\Im_{N_b}(\nu_k) \equiv \frac{1}{N_b} \sum_{q=0}^{N_b-1} \sum_{m'=M_a}^{M-1} \sum_{l'=0}^{J-1} \left| Y_{m'}(q,l') \right|^2 \tag{4.55}$$

where N_b is the number of OFDM symbols used to compute the cost function, ν_k is the frequency estimate, M_a is the number of users multiplexed in each OFDM symbol, M is the maximum number of users that can be accommodated by the system ($M > M_a$), J is the total number of sub-carriers allocated to each user, and $Y_{m'}(q,l')$ is the FFT output of

the l^{th} sub-carrier in the q^{th} OFDM symbol for user m'. Equation (4.55) sums the FFT output over the unused sub-carriers (user $m' \geq M_a$ with each user assigned J sub-carriers) for N_b OFDM symbols. The zero insertion in this algorithm is different than the zero insertion used in the Standard 802.11g circuit of Chapter III. This algorithm inserts zeros when there are not enough users to take up all the sub-carrier frequency allocations.

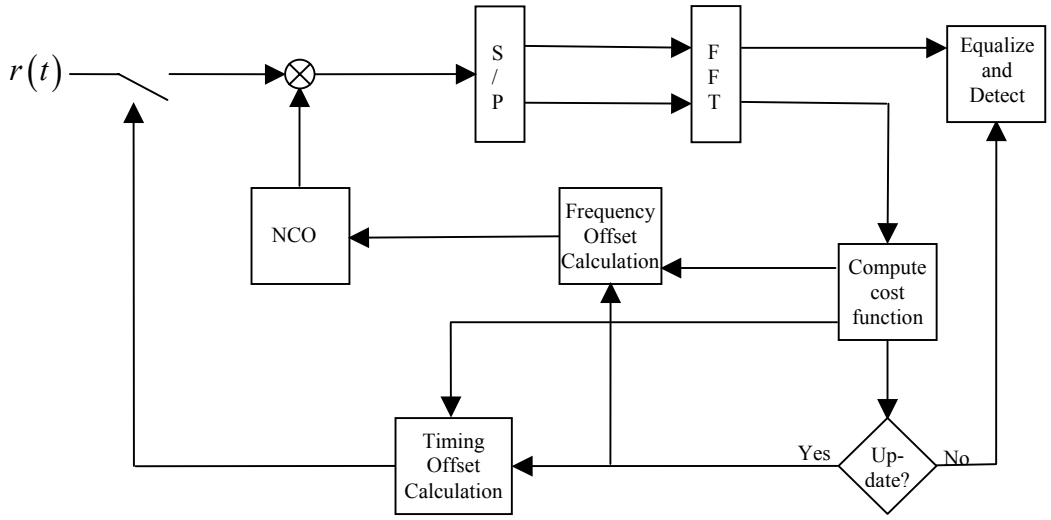

Figure 85 Virtual Carrier Synchronization Circuit [After Reference 21.]

The desired frequency is the value of ν_k such that (4.55) is minimized with respect to ν_k, or $\hat{\nu}_k = \arg\min_{\nu_k} \Im_{N_b}(\nu_k)$. The iterative process starts by setting $\hat{\nu}_k = 0$ and calculating the next frequency estimate as

$$\hat{\nu}_k(n+1) = \hat{\nu}_k(n) - \mu \frac{\delta \Im(\hat{\nu}_k(n))}{\delta \hat{\nu}_k(n)} \tag{4.56}$$

where $\hat{\nu}_k(n)$ is the carrier offset estimate at step n and μ is chosen to maximize convergence speed without loosing tracking capability.

The cost function block of Figure 85 computes the average, over a finite number of OFDM symbols, of the cost function (4.55), and its gradient and uses them to calculate a new frequency offset value from (4.56). If the new estimate exceeds a predetermined

threshold, the frequency estimate is updated. A similar method is used to calculate a timing offset. This algorithm assumes the initial frequency estimate is less than $1/2NT$, where N is the total number of sub-carriers used in each OFDM symbol and T is period of each data sample, and also makes use of a frequency-hopping scheme, where the channel allocation to each user is changed over each OFDM symbol, to average out the frequency characteristics of the channel. Because of the iterative method used in this algorithm, it is not well suited for packet-based transmission systems such as Standard 802.11g.

C. PHASE SYNCHRONIZATION

The frequency estimation and correction algorithms presented in the previous section reduce the frequency error between the receiver and the transmitter to a level that achieves acceptable signal level reception. However, even the small residual frequency error present after frequency estimation can cause the receiver to loose synchronization after only a few OFDM symbols. To compensate for these slow constellation rotations, pilot tones are inserted between the data samples to allow the receiver to track and correct for these small phase errors during the transmission. Two methods are used to estimate the frequency-shift [3]: data aided and non-data aided. The data aided technique uses the pilot tones, four $+1$ and -1s inserted in sub-carriers $-21, -7, 7,$ and 7, to estimate the phase error. The non-data aided method uses the fact that all sub-carriers are rotated the same by the phase error.

The FFT of the received data sample is given by

$$R_{nk} = H_k P_{nk} e^{j2\pi n f_\Delta T} \tag{4.57}$$

where n is the OFDM symbol number, T is the period of one OFDM symbol, k is the sub-carrier number, R_{nk} is the received pilot tone, H_k is the channel frequency response at sub-carrier frequency k, P_{nk} is the known pilot symbol, and f_Δ is the frequency offset between transmitter and receiver defined in the last section. Multiplying the received signal by an estimate given by $\hat{R}_{n,k} = \hat{H}_k P_{n,k}$ where $\hat{H}_k$ is the receiver estimate of the chan-

nel frequency response, we get $R_{n,k}\hat{R}^*_{n,k} = H_k\hat{H}^*_k\left|P_{n,k}\right|^2 e^{j2\pi nf_\Delta T}$. This can be simplified to $R_{n,k}\hat{R}^*_{n,k} = \left|\hat{H}_k\right|^2 e^{j2\pi nf_\Delta T}$ if it is assumed that the estimated channel response is perfect and noting that $\left|P_{n,k}\right|^2 = 1$. This means the phase of $R_{n,k}\hat{R}^*_{n,k}$ gives the desired phase offset. With AWGN and an imperfect estimate of the channel impulse response, the resulting noise can be averaged out with a summation.

The phase estimate from the received pilot tones of (4.57) is

$$\hat{\Phi}_n = \tan^{-1}\left\{\frac{\mathrm{Im}\left(\sum_{k=1}^{N_p} R_{nk}\cdot\left(\hat{H}_k P_{nk}\right)^*\right)}{\mathrm{Re}\left(\sum_{k=1}^{N_p} R_{nk}\cdot\left(\hat{H}_k P_{nk}\right)^*\right)}\right\} \tag{4.58}$$

where $\hat{H}_k$ is the estimated channel frequency response and N_p is the number of pilot tones used in the OFDM symbol, which is four for Standard 802.11g. [3]

This same method can be used without the use of the pilot symbols by removing the influence of the data symbols on the received signal. The received data samples after the FFT are given by

$$R_{nk} = H_k X_{nk} e^{j2\pi nf_\Delta T} \tag{4.59}$$

where X_{nk} is the k^{th} received data symbol from the n^{th} OFDM symbol.

Using the same technique as above, we estimate the resulting phase offset from (4.59) as

$$\hat{\Phi}_n = \tan^{-1}\left\{\frac{\mathrm{Im}\left(\sum_{k=-K}^{K}\left|\hat{H}_k\right|^2\left|\hat{X}_k\right|^2 e^{j2\pi nf_\Delta T}\right)}{\mathrm{Re}\left(\sum_{k=-K}^{K}\left|\hat{H}_k\right|^2\left|\hat{X}_k\right|^2 e^{j2\pi nf_\Delta T}\right)}\right\} \tag{4.60}$$

where $\hat{X}_k$ is the estimated data symbol.

If the data symbols used are BPSK modulated (as is the preamble in Standard 802.11g), then $|X_k|^2$ is known, and (4.60) can be evaluated after estimating the channel frequency response, as in (4.58).

Both of these methods require the use of an estimated channel frequency response. This estimate can be made from the training data. Standard 802.11g provides the long training symbols to help with this estimate. After the FFT, the received training symbols can be represented as

$$R_k = H_k X_k + W_k \tag{4.61}$$

where X_k are the transmitted training symbols for the k^{th} sub-carrier, H_k is the channel frequency response, and W_k is the additive noise in the channel.

The channel frequency response is estimated as

$$\begin{aligned} \hat{H}_k &= R_k \cdot X_k^* \\ &= \left(H_k X_k + W_k\right) X_k^* \\ &= H_k + W_k \cdot X_k^* \end{aligned} \tag{4.62}$$

where $|X_k|^2 = 1$ is designated by Standard 802.11g, as per the training symbol circuits used in Chapter III.

D. TIMING SYNCHRONIZATION

Timing synchronization has two aspects. The first is synchronization with the OFDM symbols, and the second is the synchronization with the data symbols within each OFDM symbol. The synchronization of the OFDM symbols requires more than just matching the symbol timing of the transmitter. The effect of timing error or jitter must also be taken into account. Figure 86 shows the effect of a symbol timing offset in the received signal.

In the top diagram, the receiver symbol timing is in exact synchronization with the transmitted symbol. Unfortunately, any variance from this timing could shift the symbol into the cyclic prefix for the next symbol, causing interference and signal degrada-

tion. In the bottom diagram, the symbol timing has been shifted into the cyclic prefix for the symbol. In this case, any variance of the symbol timing will result in a shift within the symbol's own cyclic prefix, preventing intersymbol interference. The drawback with shifting the symbol timing into the cyclic prefix is that multipath propagation in the channel could result in intersymbol interference if the timing is shifted too far into the cyclic prefix.

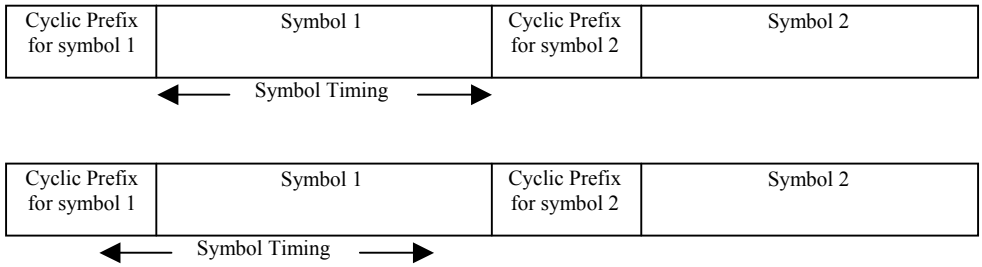

Figure 86 OFDM Nominal Symbol Timing [From Reference 3.]

Many of the algorithms used to synchronize the receiver with the transmitted carrier frequency also synchronize the receiver to the transmitted OFDM symbol. The algorithms used are the same: correlate the received signal with a known training sequence or correlate the received signal with a portion of its cyclic prefix. Using the cross-correlation between the received symbol and the known training symbols, as was done in part IV.C, we obtain

$$\hat{T}_s = \arg\max_n \left| \sum_{k=0}^{L-1} r_{n+k} t_k^* \right| \tag{4.63}$$

where $\hat{T}_s$ is the estimated OFDM symbol start time, r_n is the n^{th} received OFDM symbol, t_k is the value of the k^{th} training symbol, and L is the number of training symbols used in the calculation.

Sample clock tracking synchronizes the receiver data symbol clock with the transmitter's data symbol clock. The majority of algorithms designed to correct the sample clock tracking error use the pilot sub-carriers. Reference [25] details an algorithm that

uses this approach. The pilot tones between consecutive OFDM symbols are correlated to produce

$$Z_{l,k} = R_{l,k} R^*_{l-1,k} = \left|H_k\right|^2 \left|P_{l,k}\right|^2 e^{j2\pi k t_\Delta l \frac{T}{T_u}} \tag{4.64}$$

where $R_{l,k}$ is the FFT of the received pilot tone for OFDM symbol l at sub-carrier k, H_k is the channel impulse response for the k^{th} sub-carrier, $P_{l,k}$ is the pilot symbol for the k^{th} sub-carrier in the l^{th} OFDM symbol, $t_\Delta = \left(T_r - T_t\right)/T$ is the timing offset, where T_r is the receiver sampling period and T_t is the transmitter sampling period, T is the OFDM symbol duration, and T_u is the duration of the OFDM symbol without the guard interval.

Equation (4.64) is used for the lowest frequency pilot tone, yielding $Z_{l,k_1} = \left|Z_{l,k_1}\right| e^{j\phi_{l,k_1}}$, and the highest frequency pilot tone, yielding $Z_{l,k_2} = \left|Z_{l,k_2}\right| e^{j\phi_{l,k_2}}$. The timing estimate can then be written as

$$\hat{t}_\Delta = \frac{1}{2\pi} \cdot \frac{T_u}{T} \cdot \frac{1}{\left(k_2 - k_1\right) l} \cdot \left(\phi_{l,k_2} - \phi_{l,k_1}\right) \tag{4.65}$$

where k_1 is the smallest pilot tone index and k_2 is the largest pilot tone index.

This chapter has analyzed several techniques used to synchronize an OFDM receiver with a transmitted OFDM signal. In the next chapter packet and frequency synchronization will be simulated and compared with the theoretical results obtained in this chapter.

V. SYNCHRONIZATION PERFORMANCE ANALYSIS

With the large number of synchronization techniques available, the designation of the best performing algorithm depends on the communication channel used by the system and the type of transmitter used. This chapter evaluates the different frame and frequency synchronization techniques developed in the last chapter using System View simulation and the Standard 802.11g transmitter developed in Chapter III. All 802.11 systems are packet transmission systems; therefore, the concentration is with the data aided techniques that can obtain an accurate synchronization in the short time available with the 16-μs packet preamble. A Standard 802.11g packet construction is assumed for all the simulations.

The channel model used is the AWGN channel, simulated with two uncorrelated Gaussian noise sources and adders, which allows a comparison with the optimum zero noise channel. The two noise sources are required to simulate the transmission of the in-phase and quadrature signals in an actual Standard 802.11g system since the analysis in this thesis uses a baseband signal, i.e., the complex envelope. Although only the AWGN case is considered here, System View includes six different channel models in its communication library. In addition, the System View communications library contains parameter files that model different environments for fading channels, such as commercial, indoor and outdoor environments. This rich library of channel modeling allows the simulation of the developed circuits in many different types of environments with only small changes in the models developed here.

The block diagram of the AWGN channel model used is shown in Figure 87. The noise is zero mean with a variance that is changed to obtain channel bit error rates between 0.1 and 1×10^{-6}.

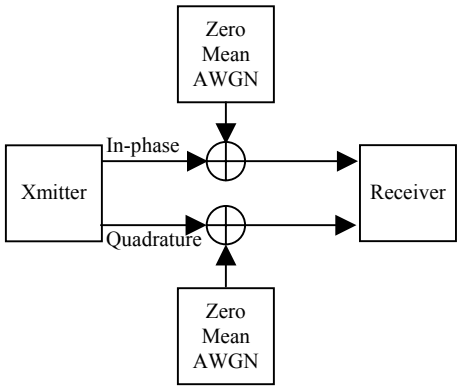

Figure 87 AWGN Simulation Block Diagram.

A. PACKET DETECTION PERFORMANCE

Two different designs are considered for packet detection. The first is a sliding-window packet detector in which the received signal is correlated with a known training sequence as introduced in Chapter IV. This not only detects a received signal, but it differentiates between non-Standard 802.11g signals and the desired signal since a correlation of the received data takes place. The second design is the delay and correlate detector discussed in Chapter IV where two adjacent sliding windows are compared to detect an incoming packet.

1. Sliding-Window Packet Detector

As discussed in Chapter IV, the sliding-window detector correlates the received data with a known sequence. The System View circuit that performs this detection is shown in Figure 88. The known sequence in this case is the Standard 802.11g short training sequence. Since the short training sequence for this standard contains 16 data samples, the length of the sliding window is also 16 samples.

The circuit of Figure 88 consists of a short training sequence generator for the transmitter, the AWGN channel, a correlation of the received signal and the locally generated (in the receiver) short training sequence, a comparator section to make a decision on the presence of a valid signal, and finally a section to compute the probability of detection. The decision variable is modified from that calculated in Chapter IV, since the known short training sequence numbers are not all identical.

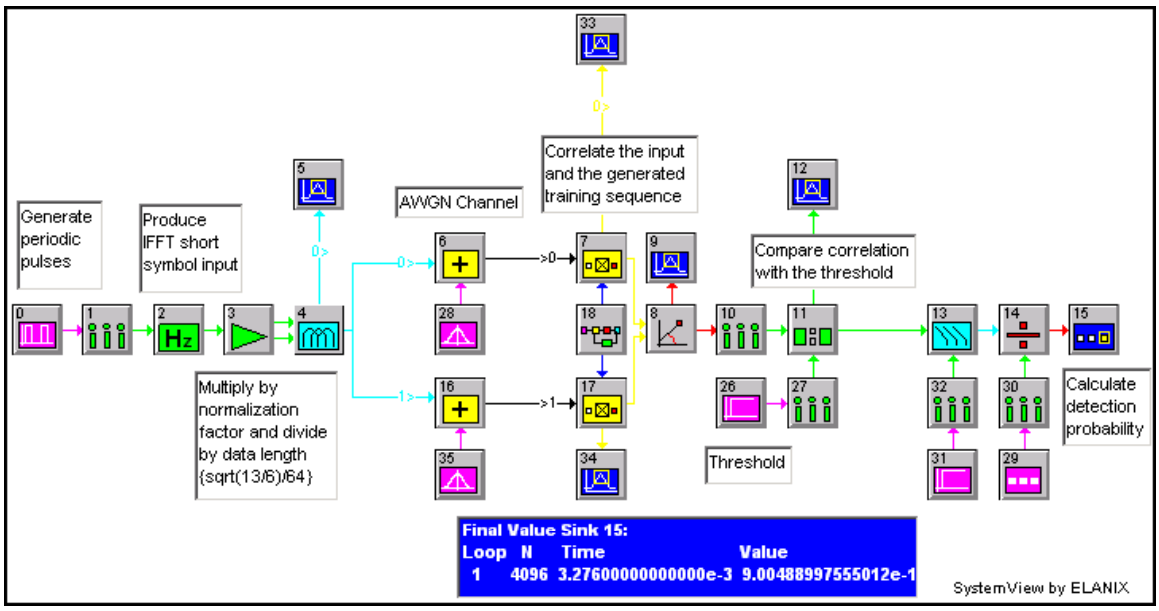

Figure 88 Sliding-Window Correlation Packet Detector.

The short training symbol sequence generator has been modified from the one used in Chapter III to allow a continuous generation of the short training symbols. The sequence generator portion of this circuit is shown in Figure 89. The pulse train generator (token 0) is used to generate the impulses needed to drive the linear filter (token 2) in the same manner as was done in Figure 44 of Chapter III. The pulse train consists of a pulse of 50 nanoseconds duration repeated every 3.2 µs. The parameters for token 0 are shown in Figure 90. Tokens one through four perform the same function as those in Figure 44. The result is an uninterrupted sequence of short training symbols, as shown in Figure 91. This uninterrupted sequence generation allows an analysis of the detection circuit performance during the first portion of the OFDM preamble. The circuit can easily be modified to conduct the same analysis over the long training symbols.

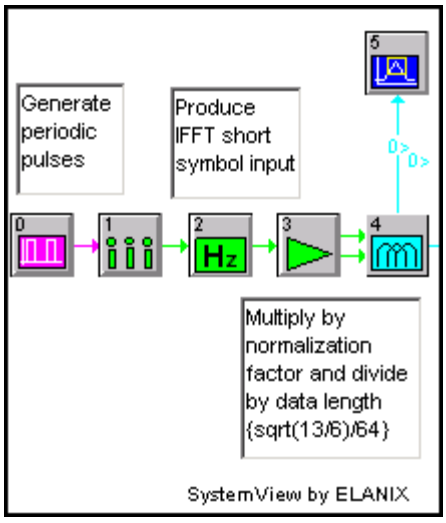

Figure 89 Uninterrupted Short Training Symbol Generation Circuit.

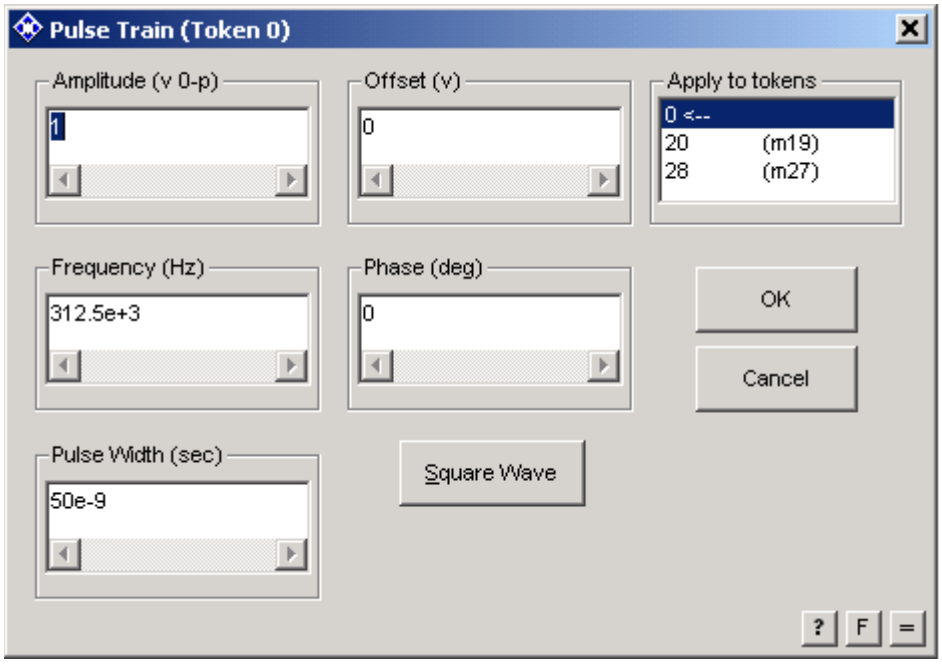

Figure 90 Pulse Train Token Parameters.

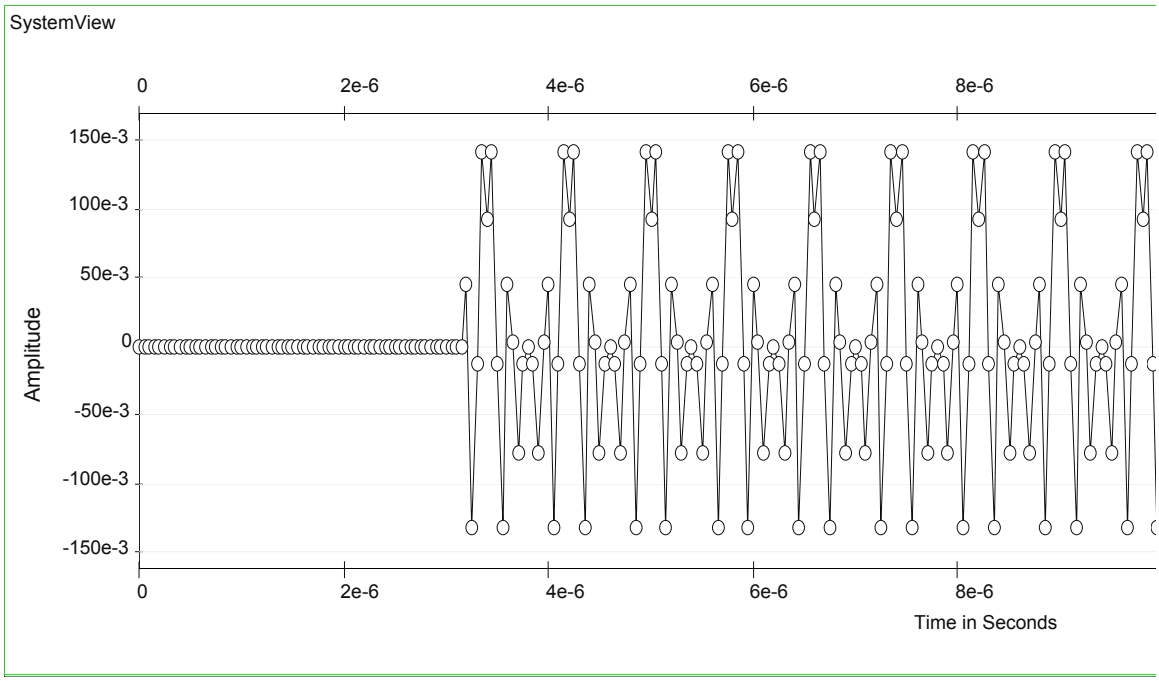

Figure 91 Uninterrupted Standard 802.11g Short Training Symbols (In-phase).

The correlation circuit is shown in Figure 92. The MetaSystem of token 18 is the local short training sequence generator of the receiver used to correlate with the received signal. The cross-correlation tokens (token seven and 17) perform the cross-correlation of the locally generated short training sequence with the received signal. The parameters of these tokens are shown in Figure 93. The summation in (4.17) can be modeled using the cross-correlation of the received signal and the generated training symbols, where the summation corresponds to the position where the two signals are aligned in time, yielding maximum correlation. The signal received on the in-phase channel is correlated with the in-phase training sequence, and the signal received on the quadrature channel is correlated with the quadrature training sequence. The final result is converted into a magnitude and compared to the threshold. When the received signal and the training symbols are aligned in time, (4.17) for the in-phase signal is

$$\begin{aligned}\sum_{m=1}^{k} r_{mi} s_{mi} &= \sum_{m=1}^{k} \left(s_{mi} + n_{mi} \right) s_{mi} \\ &= \sum_{m=1}^{k} s_{mi}^2 + \sum_{m=1}^{k} s_{mi} n_{mi}\end{aligned} \tag{5.1}$$

where s_{mi} is the in-phase component of the training symbol and n_{mi} is the in-phase channel noise. Similarly, the equivalent of (5.1) for the quadrature signal can be written as

$$\begin{aligned}\sum_{m=1}^{k} r_{mq} s_{mq} &= \sum_{m=1}^{k} \left(s_{mq} + n_{mq} \right) s_{mq} \\ &= \sum_{m=1}^{k} s_{mq}^2 + \sum_{m=1}^{k} s_{mq} n_{mq}\end{aligned} \tag{5.2}$$

where s_{mq} is the quadrature component of the short training symbol and n_{mq} is the quadrature channel noise. Since the noise in both channels is Gaussian with zero mean and variance σ_n^2, the correlations of (5.1) and (5.2) for each channel are also Gaussian with mean $\sum_{m=1}^{k} s_{mi}^2 = \sum_{m=1}^{k} s_{mq}^2$ and variance $\sigma_n^2 \sum_{m=1}^{k} s_{mi}^2 = \sigma_n^2 \sum_{m=1}^{k} s_{mq}^2$.

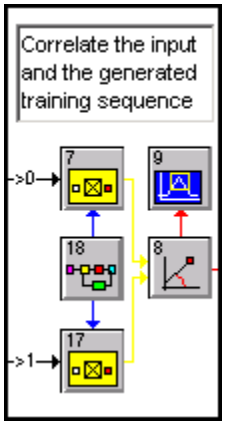

Figure 92 Sliding-Window Correlation Circuit.

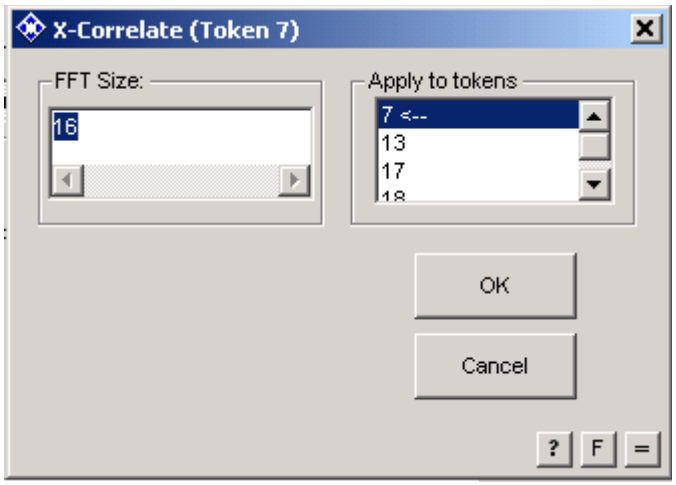

Figure 93 Cross-Correlation Token Parameters.

The comparator section of the packet detection circuit of Figure 88 is shown in Figure 94. Token nine converts the complex signals received from the adders into polar coordinates, i.e., a magnitude and phase. In the presence of the AWGN in the channel, this token converts the Gaussian distributed signals at the outputs of the adders to a Ricean distributed variable for the magnitude output of the token when a packet is present and a Rayleigh distributed variable when no packet is present. This output is then compared with a fixed threshold (token 35) to determine if a packet has been detected.

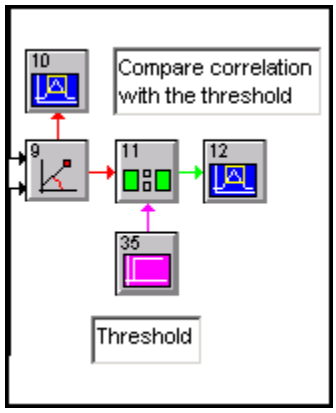

Figure 94 Comparator Circuit.

To compute the desired threshold, the probability distribution of the magnitude output from token eight of Figure 88 must be determined under the two hypotheses of (4.15). If X is defined as the Gaussian random variable from the output of the in-phase correlation token seven of Figure 88 and Y is defined as the Gaussian random variable from the output of the quadrature correlation token seventeen of Figure 88, then the magnitude output of token eight is $R = \sqrt{X^2 + Y^2}$, if a signal is present. The random variable R is a Ricean distributed random variable with a distribution function of

$$p_R(r) = \begin{cases} \dfrac{r}{\sigma^2} e^{-\frac{\left(r^2+s^2\right)}{2\sigma^2}} I_0\left(\dfrac{rs}{\sigma^2}\right) & \text{if } r \geq 0 \\ 0 & \text{if } r < 0 \end{cases} \tag{5.3}$$

where $\sigma^2 = \sigma_n^2 \sum_{m=1}^{k} s_{mi}^2 = \sigma_n^2 \sum_{m=1}^{k} s_{mq}^2$ is the variance of the random variables X and Y, $s^2 = m_X^2 + m_Y^2$, and I_0 is the modified Bessel Function of the first kind and order zero defined as $I_0(x) = \sum_{k=0}^{\infty} \left(\left[\left(x/2 \right)^{2k} \right] \Big/ \left[(k!)^2 \right] \right)$ where $x \geq 0$. A graph of (5.3) is shown in Figure 95 where the noise variance has been set at $\sigma_n^2 = 1$, resulting in $\sigma^2 = (1) \sum_{m=1}^{16} s_{mi}^2 = 0.10156$, and $s = \sqrt{\sum_{m=1}^{16} s_{mi}^2 + \sum_{m=1}^{16} s_{mr}^2} = 0.14363$ for the Standard 802.11g short training symbols of Figure 91. The Gaussian distribution of the input variable X and Y is shown as the dashed blue line and the Rayleigh distribution (i.e., (5.3) with $s = 0$) of X and Y is shown as the dashed black line for comparison.

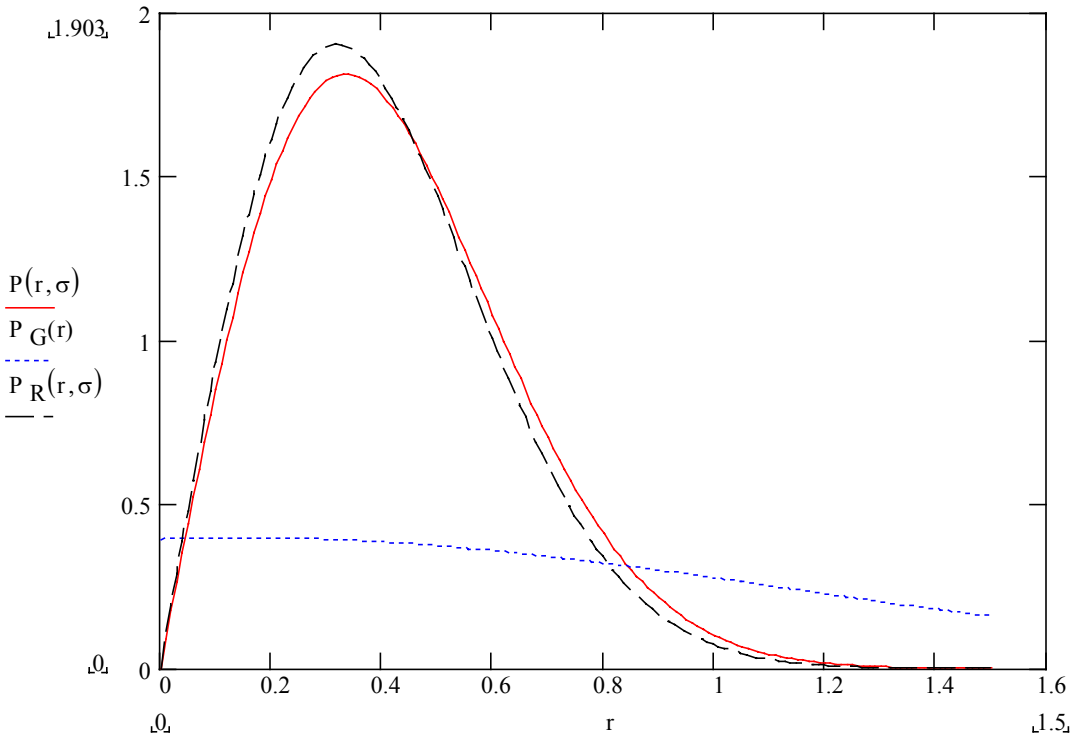

Figure 95 Decision Variable Probability Distribution.

Combining (5.3) with Equations (4.4) and (4.12) that define the probability of false alarm and the probability of detection, we obtain the probability of detection as

$$P_D = \int_{\eta}^{\infty} \frac{r}{\sigma^2} e^{-\frac{\left(r^2+s^2\right)}{2\sigma^2}} I_0\left(\frac{rs}{\sigma^2}\right) dr \tag{5.4}$$

where $s^2 = (0.14363)^2$ and $\sigma^2 = \sigma_n^2 (0.10156)$ for the Standard 802.11g short training sequence.

The probability of false alarm is given by

$$P_{FA} = \int_{\eta}^{\infty} \frac{r}{\sigma^2} e^{-\frac{\left(r^2+s^2\right)}{2\sigma^2}} I_0\left(\frac{rs}{\sigma^2}\right) dr \tag{5.5}$$

where $s^2 = 0$ and $\sigma^2 = \sigma_n^2 (0.10156)$. Since $s = 0,$ and $I_0(0) = 1,$ (5.5) reduces to

$$P_{FA} = \int_{\eta}^{\infty} \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} dr = -e^{-\frac{r^2}{2\sigma^2}} \Big|_{\eta}^{\infty} = e^{-\frac{\eta^2}{2\sigma^2}}. \tag{5.6}$$

Solving (5.6) for the threshold η, we get $\eta = \sqrt{2\sigma^2 \ln\left(1/P_{FA}\right)}$.

Equation (5.4) has the solution

$$P_D = Q_0\left(\frac{s}{\sigma_n}, \frac{\eta}{\sigma_n}\right) \tag{5.7}$$

where $Q_0(a,b)$ is Marcum's Q-function and is defined as

$Q_0(a,b) = \int_b^{\infty} x e^{-\left(x^2+a^2\right)/2} I_0(ax)\, dx$. Equation (5.7) was solved numerically using the Mathcad 2001 Professional program, as shown in Appendix B. To obtain a probability of detection of $P_D = 0.9000$ requires a threshold of $\eta = 0.1539$ under the conditions of Figure 95. This threshold value results in a probability of false alarm of $P_{FA} = 0.8900$.

The value of $\eta = 0.1539$ was used in the circuit of Figure 88. Running the program five times with a total of 4096 possible detections for each trial with a noise variance of $\sigma_n^2 = 1$, the probability of detection averaged 0.9003. Using these values in (5.6) resulted in a calculated $P_{FA} = 0.8900$. Figure 88 was also modified to change the short training symbol input to zero, and the simulation was run again five times with 4096 trials per run. The result was a $P_{FA} = 0.8930$, close to the predicted value of 0.8900. This seems like a large false alarm probability, but it is more the result of the choice of noise variance used in the calculation than representative of the performance of the algorithm. If the noise variance is changed to $\sigma_n^2 = 0.1$ and the probability of detection is kept constant at 0.9000 (this changes the threshold to $\eta = 0.0750$), the numerical evaluation of the threshold gives a probability of false alarm of 0.7579. At $\sigma_n^2 = 0.01$ with the probability of detection kept at 0.9000 (the threshold in this case is $\eta = 0.1069$), the probability of false alarm falls to 3.5955×10^{-3}. This significant improvement in the probability of false alarm can be seen in Figure 96 where the distribution for the probability of false alarm is

shown with the black dashed line and the probability of detection is shown with the red solid line. The separation between the two curves caused by the reduction in the AWGN variance, σ_n^2, reduces the probability of false alarm for a given detection threshold, η.

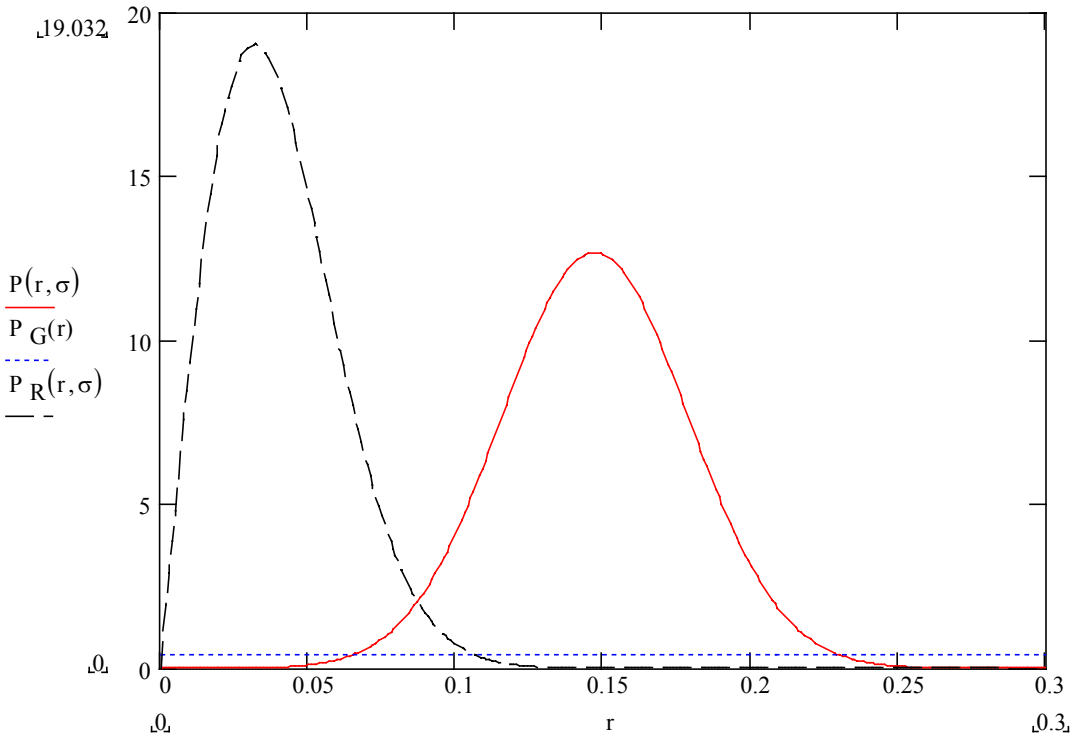

Figure 96 Decision Variable Distribution with Reduced AWGN.

2. Delay and Correlate Packet Detector

The System View circuit used to perform the delay and correlate packet detection simulation of the AWGN channel is shown in Figure 97, which is an implementation of the block diagram of Figure 78.

To find the detection threshold for this circuit, a Gaussian distribution in the form of (4.28) is assumed for the decision statistic (4.27). The mean value of this distribution is given by (4.29) and the variance is given by (4.30). The σ_n^2 used in (4.29) is the channel noise variance. The signal power $\overline{s}^2$ is defined by Reference [17] as $\overline{s}^2 = E\left\{\left[\mathrm{Re}\left(s_m\right)\right]^2\right\}$

and $\overline{s}^2 = E\left\{\left[\mathrm{Im}\left(s_m\right)\right]^2\right\}$. For the Standard 802.11g training samples, $E\left\{\left[\mathrm{Re}\left(s_m\right)\right]^2\right\} = (1/L)\sum_{m=0}^{L-1} s_m^2$ and with $L = 16$, $E\left\{\left[\mathrm{Re}\left(s_m\right)\right]^2\right\} = 0.1015625/16$ or $\overline{s}^2 = 6.3477\times10^{-3}$.

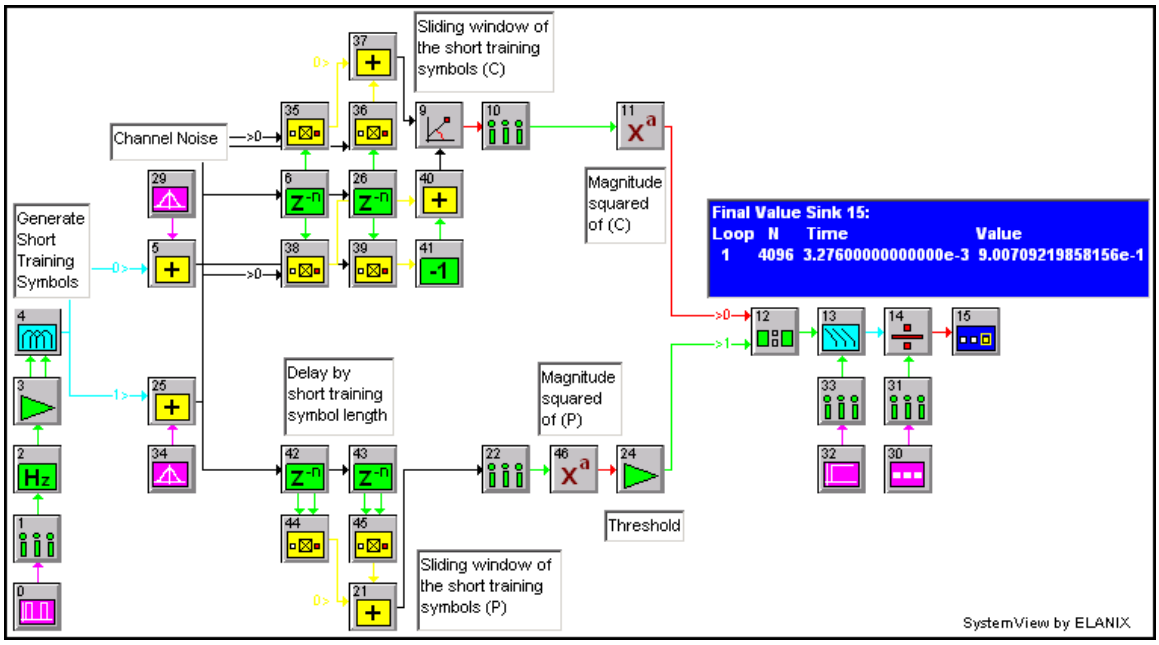

Figure 97 Delay and Correlate System View Circuit.

With no noise input, the signals generated by the *C*-sliding window portion of the circuit are equal to the signals generated by the *P*-sliding window portion of the circuit when the short training symbols are being received. This is seen by examining (4.27) and realizing that in the case of training symbols received with no noise, $r_{n+k} = r_{n+k-D}$ due to the repetition of the training symbols. Therefore, in this case, $C = P$. If noise is present, but no short training symbols are being received, then the *P*-sliding window portion of the circuit generates a significant signal because of the zero lag correlation, but the *C*-sliding window portion of the circuit only generates a small signal since the noise data is uncorrelated and the correlation lag is non-zero. When training symbols are received, the *C*-sliding window signal increases because of the correlation of the repeated training symbols.

The circuit of Figure 97 has a similar construction to the circuit of Figure 88. The first section consists of a continuous short training symbol generation circuit. Tokens 5, 25, 29, and 34 are the Gaussian noise channel. The cross-correlation and adder tokens 35 - 41 provide the *C*-sliding window and $21,\ 42$ - 45 the *P*-sliding window calculations. The correlations are sampled by tokens 10 and 22 every 16 clock cycles and compared to determine if a detection has occurred. The output of the comparator (token 12) is connected to a probability of detection calculation circuit, as performed in the previous circuit.

The threshold of token 28 is calculated using the Gaussian distribution of (4.28). The mean value of the distribution, using (4.29) and a noise variance of $\sigma_n^2 = 1\times10^{-5}$ is

$$\mu = \frac{\left(6.3477\times10^{-3}\right)^2}{\left(6.3477\times10^{-3}+1\times10^{-5}\right)^2} = 0.9969 \tag{5.8}$$

The noise variance is reduced significantly to allow for a Gaussian approximation. The previously used noise variance value of $\sigma_n^2 = 1$ gives a poor prediction of the required threshold. The decision statistic's variance, given by (4.30), is

$$\sigma^2 = \frac{2\left(6.3477\times10^{-3}\right)^2\left[\left(1.9969\right)\left(6.34\times10^{-3}\right)\left(10^{-5}\right)+\left(1+2\left(.9969\right)\right)\left(10^{-5}\right)^2\right]}{16\left(1\times10^{-5}+6.3477\times10^{-3}\right)^4} = 3.9168\times10^{-4} \tag{5.9}$$

Using the mean and variance given in (5.8) and (5.9), we now calculate the threshold to obtain a probability of detection of $P_D = 0.9000$ to be $\eta = 0.9964$. Simulation using this threshold yielded 4096 detections in 4096 trials. The threshold value that actually obtains an average $P_D = 0.9$ in the simulation with the number of trails at 4096 in the circuit of Figure 97 is $\eta = 0.999918$. It is conjectured that this discrepancy is attributable to the Gaussian approximation.

B. FREQUENCY SYNCHRONIZATION PERFORMANCE

The circuits built to test the frequency synchronization algorithms of Chapter IV are split up into the frequency-domain and the time-domain. The biggest difference between the two is the frequency resolution of the algorithms. In the frequency-domain (i.e., after the receiver FFT) the separation between repeated training symbols is 64 clock cycles, whereas in the time-domain (i.e., before the receiver FFT), the separation is 16 clock cycles for the short training symbols and 64 clock cycles for the long training symbols. As discussed in Chapter IV and from (4.54), this results in a maximum frequency resolution of either 625 kHz for a separation of 16 samples or a maximum frequency resolution of 156.25 kHz with a separation of 64 samples.

1. Frequency-Domain Frequency Synchronization

The System View circuit diagram showing frequency synchronization using the correlation between the data in adjacent OFDM symbols to estimate the frequency error is shown in Figure 98.

Tokens zero through four are used to generate the short training sequence, the same as for the packet detection systems of Figure 97 and Figure 88. However, with this circuit the system clock is set at 4.8 GHz, which is twice the carrier frequency of 2.4 GHz. This allows the analysis of the carrier frequency synchronization. The sampler token (token one) is used to step down the pulse train (token zero) to the required frequency of 312.5 kHz. The resample tokens (tokens five and 39) are used to change the sample frequency to 4.8 GHz from the 20 MHz used to generate the short training pulses, allowing correct operation with the multiplier tokens (tokens 6 and 40). The short training symbols are then modulated with 2.4-GHz in-phase and quadrature carriers (token 42) and sent to the receiver through an AWGN channel.

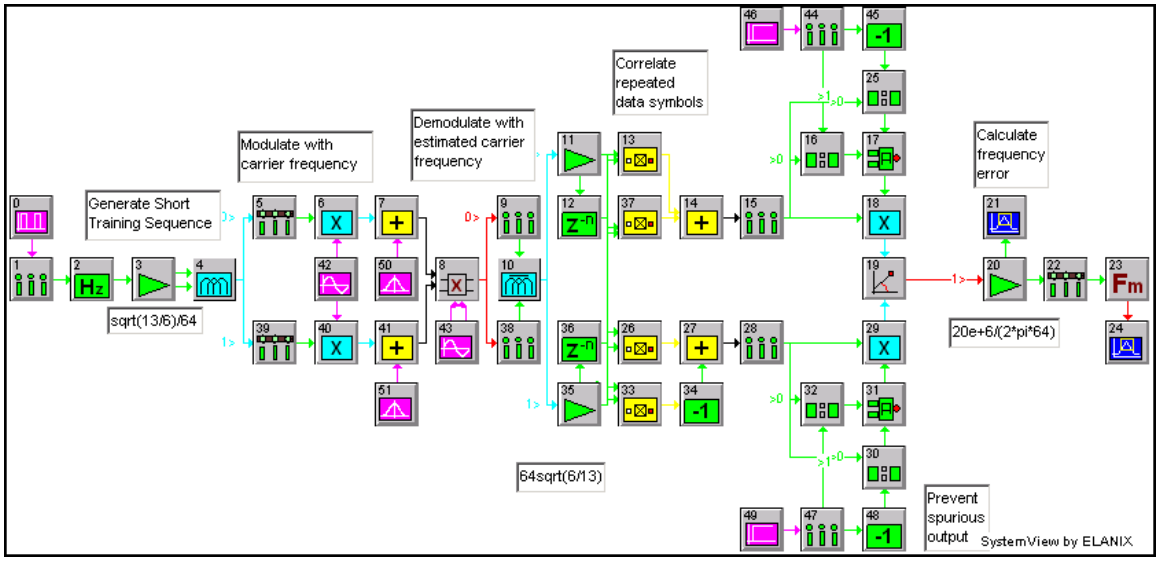

Figure 98 Frequency-Domain Frequency Synchronization.

At the receiver, the signal is demodulated with the complex signal $e^{-j2\pi f_r nT}$ by token eight. The resulting signal is then stepped down to the data rate of 20 MHz with the sampling tokens (tokens nine and 38). At the output of the OFDM demodulator, token 10, are the short training symbols, which are repeated each OFDM symbol. Since the transmitter and receiver carrier frequencies are offset, these symbols should only be different from each other by a phase offset proportional to the difference in frequencies of the transmitter and receiver if we neglect the noise.

The gain tokens (tokens 11 and 35) remove the normalization factor used in the transmitter. The correlation of the received symbols with the symbols delayed by one OFDM symbol is accomplished by the cross-correlator tokens 13, 26, 33, and 37. The correlation of the two symbols is given by

$$\begin{aligned}\sum_{n=1}^{64} r_n r_{n-D}^{*} &= \sum_{n=1}^{64} \left(r_{nr} + jr_{ni}\right)\left(r_{(n-D)r} + jr_{(n-D)i}\right)^{*} \\ &= \left[\sum_{n=1}^{64} r_{nr} r_{(n-D)r} + \sum_{n=1}^{64} r_{ni} r_{(n-D)i}\right] + j\left[\sum_{n=1}^{64} r_{ni} r_{(n-D)r} - \sum_{n=1}^{64} r_{nr} r_{(n-D)i}\right].\end{aligned} \tag{5.10}$$

The summations of (5.10) are implemented by tokens 13, 37, 26, and 33 respectively. The delay tokens are set to 64 samples (the number of samples per OFDM symbol, i.e., $D = 64$).

The sampler tokens (tokens 15 and 28) at the output of the adder tokens sample the summation output once every OFDM symbol (312.5 kHz). This error signal is converted from Cartesian coordinates into polar coordinates by token 19. The phase of the error signal is then converted to a frequency error by the gain token (token 20) by dividing by $2\pi NT$ where $N = 64$ and $T = 50$ ns.

The frequency error is resampled by token 22 to convert it to the system sample rate. This is fed into the input of the frequency modulator token (token 23). The parameters used in this token are shown in Figure 99. The token frequency has been deliberately set to 100 kHz below the transmitter frequency to evaluate the effect of a 100-kHz frequency offset. The output from the frequency modulator can then be used as the system clock to demodulate the signal used by the receiver circuit or as part of a feedback circuit used in a phase-locked loop.

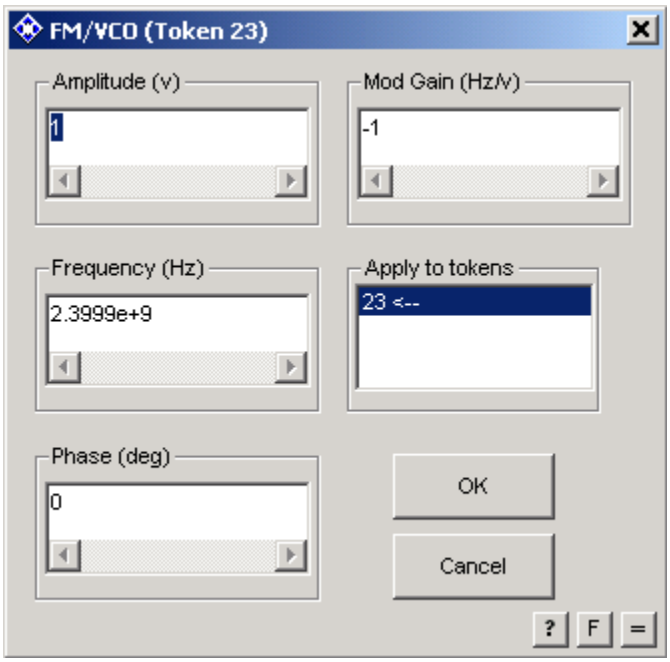

Figure 99 Frequency Modulation Token Parameters.

The error signal produced by this circuit for a 100-kHz receiver frequency offset with no channel noise is shown in Figure 100. As shown, the frequency error is detected and corrected 12.8 µs into the run, which is 9.6 µs (3.2 µs each for the OFDM demodu-

lator, the sample delay, and correlation token) after the signal has been received. This delay is due to the OFDM modulation and demodulation processes and the cross-correlation, all of which take $3.2\ \mu s$ to calculate their outputs.

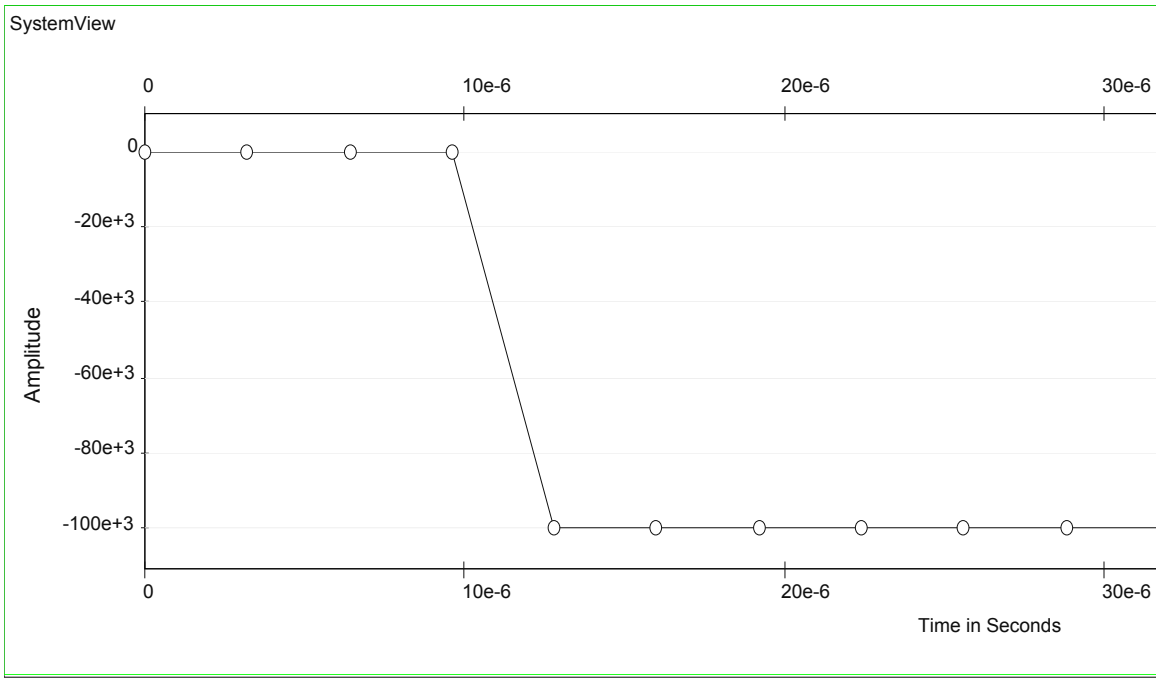

Figure 100 Frequency-Domain Frequency Error.

The output of the frequency modulator is shown in Figure 101. The beat frequency output for the first $12.8\ \mu s$ is caused by the difference in the receiver frequency and the received frequency. Since the received frequency is 2.4 GHz and the receiver frequency is 2.3999 GHz, the difference is 100 kHz. As can bee seen from Figure 101 the period of the beat frequency is $10\ \mu s$ corresponding to a frequency difference of 100 kHz. The frequency of the voltage-controlled oscillator is stable at the received frequency after the $12.8\ \mu s$ required for the circuit to begin working. However, the output is attenuated from the expected value of one volt to a value of 0.1872 volts. This corresponds to an offset in phase from the received signal by $\cos^{-1}(0.1872) = 79.2^{\circ}$. Sampling the received signal at times other than the maximum and minimum voltage levels causes this

phase offset. This also demonstrates the requirement for a phase synchronization circuit after frequency synchronization has been accomplished.

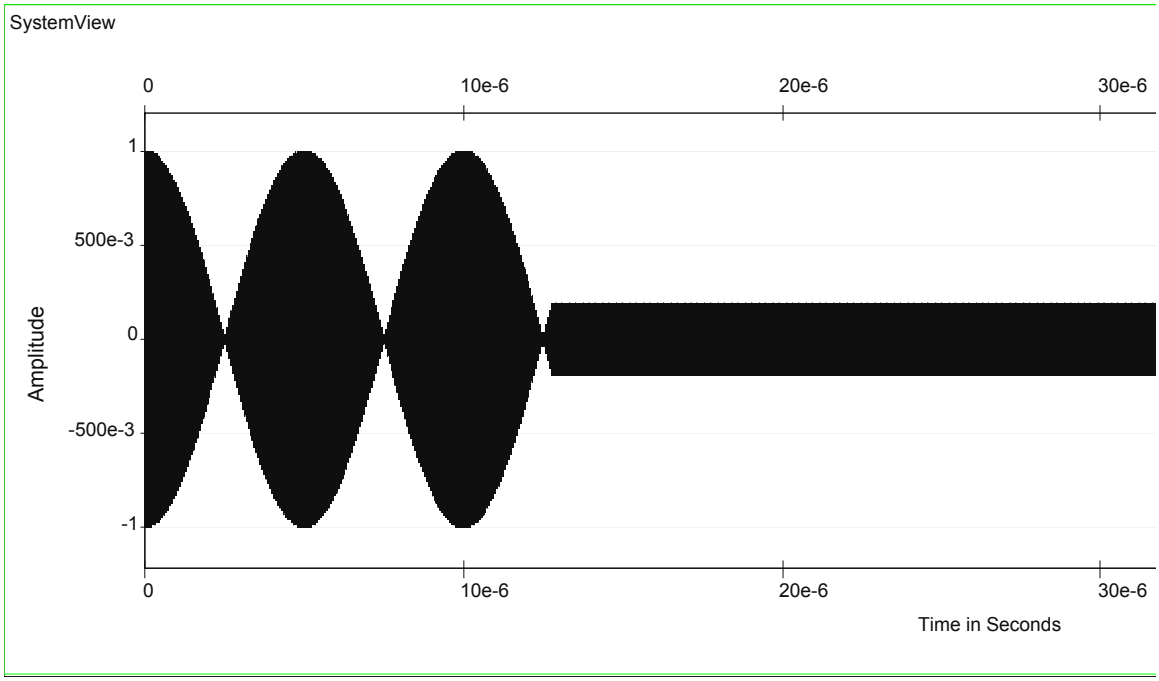

Figure 101 Frequency-Domain Synchronizer Receiver Frequency.

The effect of noise in the circuit is demonstrated in Figure 102 and Figure 103. The same frequency offset of 100 kHz was used with a noise standard deviation of $\sigma_n = 0.01$. The bit energy-to-noise power spectral density in this circuit can be calculated from the average short training symbol energy of the transmitted training symbol divided by the noise power spectral density as shown in (5.11):

$$\frac{E_b}{N_0}[\text{dB}] = 10\log\left(\frac{\text{Average bit energy}}{\text{Noise power spectral density}}\right) \tag{5.11}$$

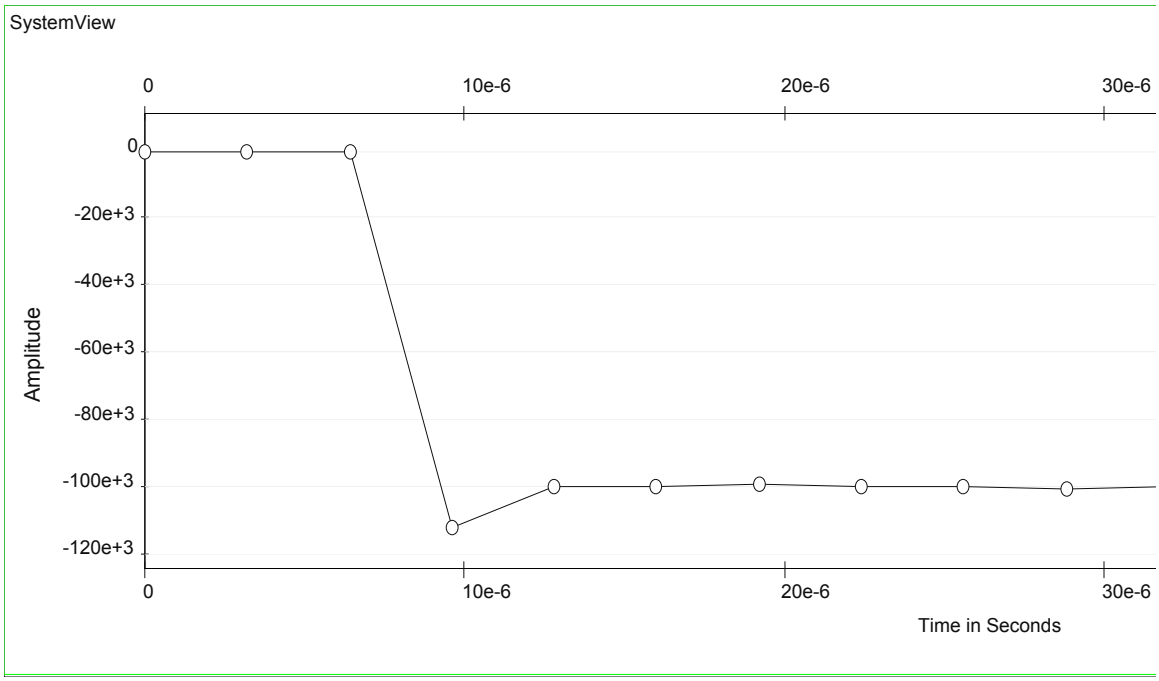

Figure 102 Frequency Synchronization Error Performance with AWGN.

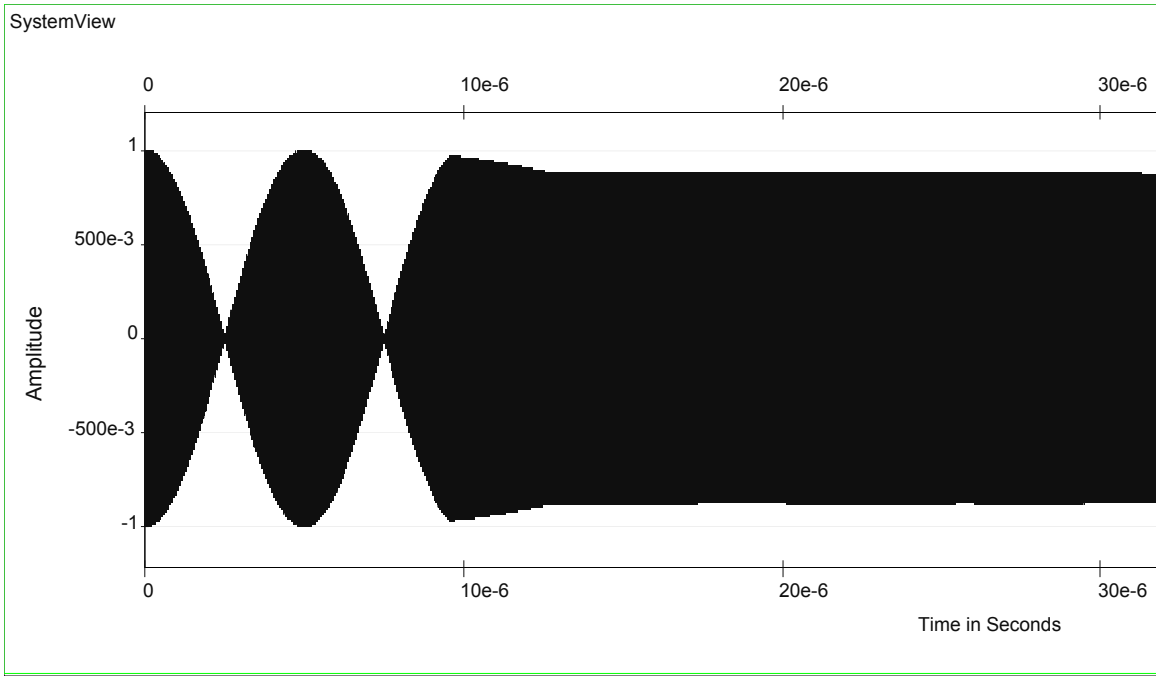

Figure 103 Frequency Synchronization Frequency Performance with AWGN.

The average signal power for the Standard 802.11g short training samples was calculated in Section A.2 as $\overline{s}^2 = 6.3477 \times 10^{-3}$. This means the average bit energy is $E_{b_{avg}} = \overline{s}^2 T_b$ where T_b is the duration of one bit. Since $T_b = 50 \times 10^{-9}$ sec for the transmitted Standard 802.11g short training samples, this results in an average bit energy of $E_{b_{avg}} = 3.1739 \times 10^{-10}$ J. Since $N_0 = \sigma_n^2 / W$ where $W = 16.6$ MHz is the bandwidth of the Standard 802.11g transmitted signal, the bit energy-to-noise power spectral density of this simulation is

$$\frac{E_b}{N_0}[\text{dB}] = 10\log\left(\frac{E_b}{N_0}\right) = 10\log\left(\frac{\left(6.3477 \times 10^{-3}\right)\left(50 \times 10^{-9}\right)}{\left(0.01\right)^2 \Big/ \left(16.6 \times 10^6\right)}\right) = 17.2169 \text{ dB}. \quad (5.12)$$

Equation (5.12) shows that a fairly large bit energy-to-noise power spectral density ratio is required to track the incoming signal in this simulation and provide for a reasonably stable output.

2. Time-Domain Frequency Synchronization

The circuit used for the time-domain frequency synchronization is shown in Figure 104. This circuit is closely modeled after the frequency-domain synchronizer of Figure 98, with the exception of the lack of OFDM demodulation and the delay of 16 samples used in the correlation instead of the 64 sample delay used in the previous circuit. This difference in samples used is caused by the time it takes for the timing samples to repeat themselves. The timing samples transmitted by the Standard 802.11g transmitter repeat every 16 samples, as shown in Figure 48 in Chapter II. Once the received training samples are demultiplexed by the FFT, the training samples repeat only every 64 samples.

The error signal output for this circuit with no noise input is shown in Figure 105. The frequency of the receiver was set to 100 kHz below the frequency of the transmitter, as was done previously in Figure 100. The frequency error converges to the correct error of 100 kHz at 4.8 μs which is 1.6 μs after the signal is received. This is much quicker

than the 9.6 μs required by the frequency-domain synchronizer and well within the short training OFDM symbol duration of 8 μs.

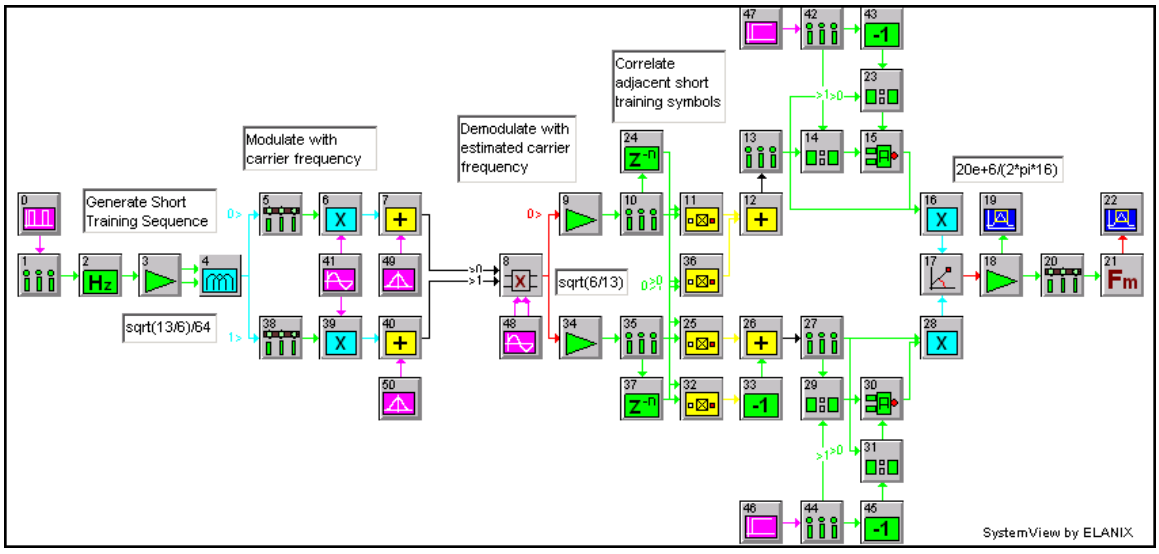

Figure 104 Time-Domain Frequency Synchronization Circuit.

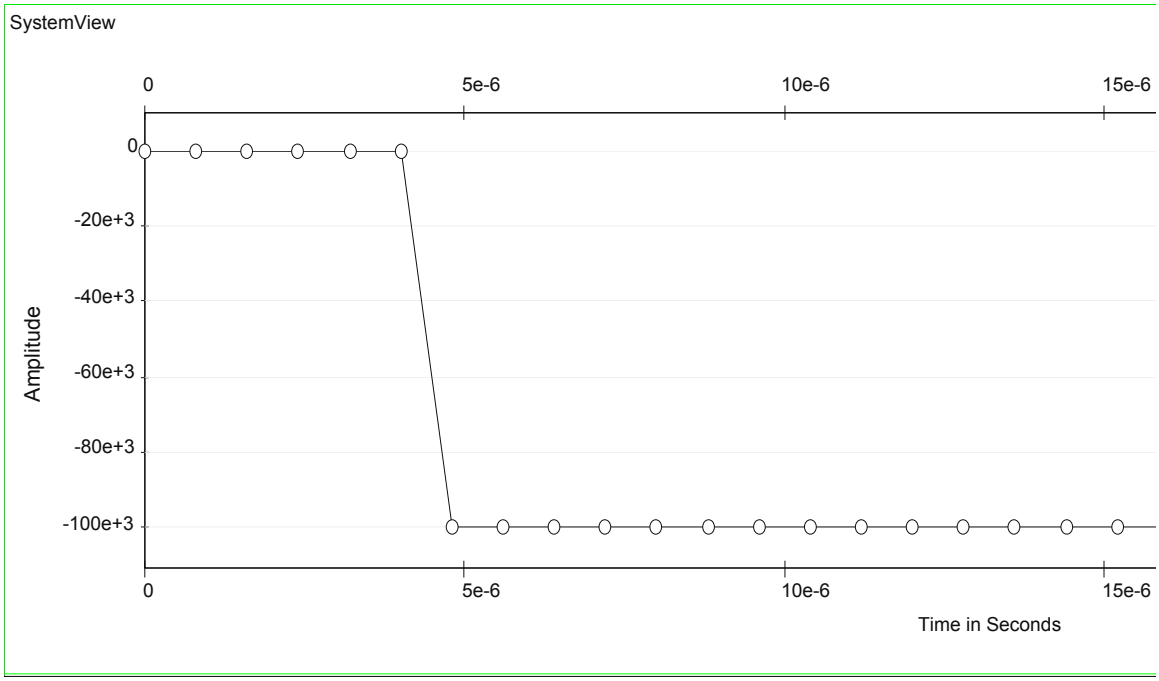

Figure 105 Time-Domain Frequency Error.

The output of the voltage-controlled oscillator token for this circuit is shown in Figure 106. Again, the beat frequency output for the first $4\ \mu s$ is caused by the difference in the receiver frequency and the received frequency. Since the received frequency is 2.4 GHz and the receiver frequency is 2.3999 GHz, the difference is 100 kHz. As can be seen from Figure 105, the period of the beat frequency is $10\ \mu s$ corresponding to a frequency difference of 100 kHz. The frequency of the voltage-controlled oscillator is stable at the received frequency after the $4\ \mu s$ required for the circuit to begin working. However, the output is attenuated from the expected value of one volt to a value of 0.9921 volts. This corresponds to an offset in phase from the received signal by $\cos^{-1}(0.9921) = 7.2^{\circ}$. Sampling the received signal at times other than the maximum and minimum voltage levels causes this phase offset.

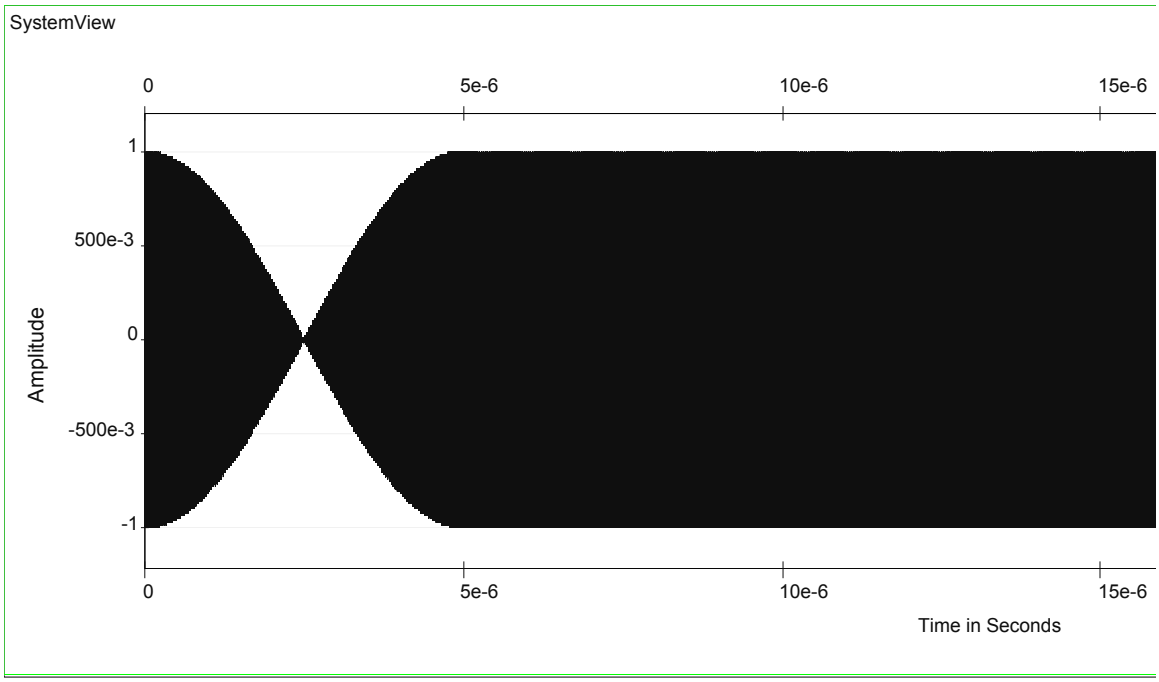

Figure 106 Time-Domain Synchronizer Receiver Frequency.

The performance of this circuit with AWGN is better than the performance of the frequency-domain synchronizer. The time-domain frequency synchronization with a noise standard deviation of $\sigma_n = 0.1$ resulted in a widely varying output frequency. At a

noise deviation of $\sigma_n = 0.02$ the output was more stable. The error signal for this case is shown in Figure 107 and the voltage-controlled oscillator output is shown in Figure 108. This value of noise deviation corresponds to a bit energy-to-noise power spectral energy ratio of $E_b/N_0 = 11.1963$ dB, giving a better performance in a noisy environment.

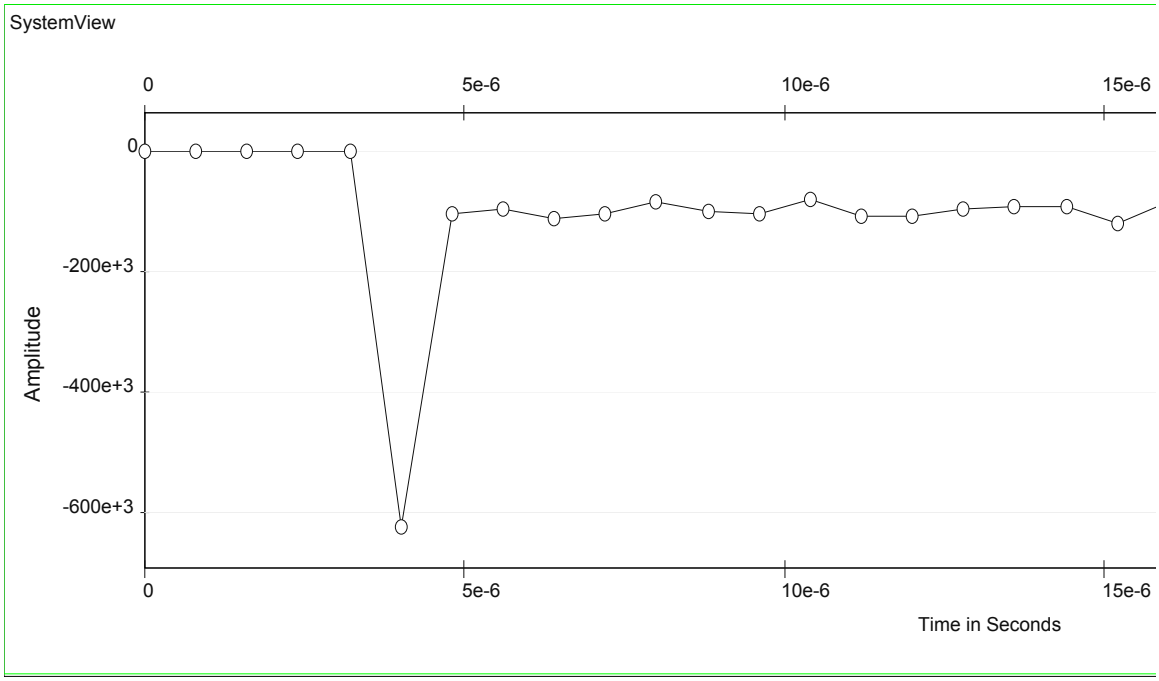

Figure 107 Time-Domain Synchronizer Frequency Error in AWGN.

The time-domain synchronization circuit achieves synchronization faster than the frequency-domain synchronizer, and the time-domain synchronizer gives a stable output frequency at a lower bit energy-to-noise energy ratio under AWGN conditions. The faster synchronization of the time-domain circuit is expected, since the time-domain circuit does not perform an FFT on the data before correlation of the received signal. However, the better performance under AWGN conditions was not something predicted before running the simulation. It is conjectured that the synchronization error present while performing the FFT in the frequency-domain circuit results in a loss of orthogonality between the sub-carriers and increased interference between the sub-carriers, resulting in reduced performance.

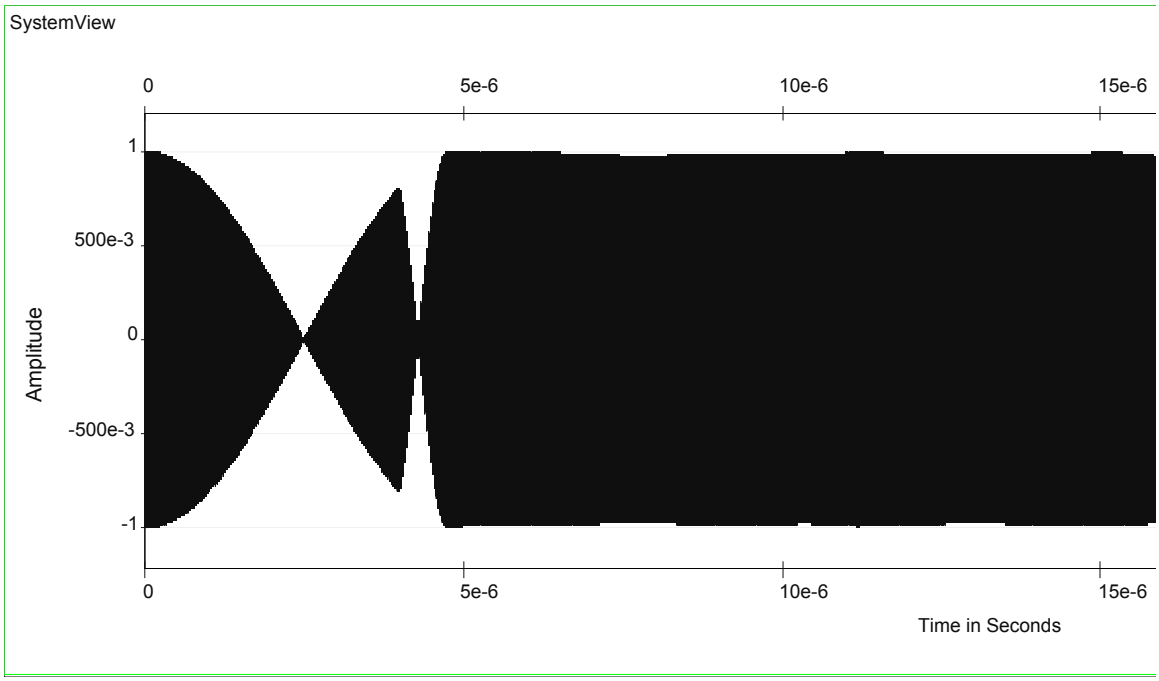

Figure 108 Time-Domain Synchronizer Carrier Output in AWGN.

This chapter has implemented and analyzed some of the circuits developed in Chapter IV and showed how the theory matches with the practical implementation of the circuit. The next chapter will summarize this thesis and recommend further areas of research required.

VI. CONCLUSIONS

The goal of this thesis was to describe the methods that can be used to provide synchronization in an OFDM system and then use these methods in the design of actual circuits. As described in Chapter IV, there are many algorithms available to synchronize OFDM systems. This thesis focused on the algorithms useful for a packet-based system to limit its scope. However, even with this limited scope, this thesis was limited to frame and carrier frequency synchronization in an AWGN environment. Even so, there are enough different algorithms available that not all can be adequately covered in a single thesis. In addition to the small number of synchronization algorithms considered, the channel environment was limited to the AWGN case, even though most OFDM systems are used in an environment where multipath is a significant problem and the channel characteristics can change frequently, as in a mobile environment. However, the concepts and circuit designs used in this thesis can easily be extended to the multipath, fading channel environment.

A. FINDINGS

Although the performance of the different algorithms analyzed in an AWGN environment were satisfactory, a common theme emerged in the synchronization of OFDM signals. The received signal must be correlated with some known data to obtain easily an error signal. This was true for both packet-based systems and continuous transmission systems. The unique construction of the Standard 802.11g OFDM symbol using a cyclic prefix and virtual sub-carriers allows use of the known data to get optimum synchronization results.

Based on the analysis in Chapter IV and the simulation discussed in Chapter V, the best performing algorithm discussed in this thesis for frame synchronization was the sliding-window correlator packet detector illustrated in Figure 88 since it provided packet detection with low probability of false alarm with a noise variance of $\sigma_n^2 = 0.1$. The delay and correlate packet detector illustrated in Figure 97 appears to have the potential for good performance. However, the Gaussian approximation did not accurately predict the

desired threshold. With further analysis, a better detection probability may allow a threshold detection prediction that will allow a comparison with the sliding-window correlator.

The best performing carrier frequency synchronization algorithm analyzed in this thesis was the time-domain frequency synchronizer illustrated in Figure 104. Its improved performance in AWGN and its ability to achieve synchronization in a shorter time compared to the frequency-domain synchronizer illustrated in Figure 98 make it the better performing circuit.

System View simulation proved to be a useful tool, not only in conceptualizing the practical design of the required circuits, but also in verifying the analytical solutions derived in Chapter IV. Although the software can be challenging to learn and use, its analysis capability is a strong reason to use this software during initial circuit analysis and design.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

Due to the breadth of the synchronization subject area and the dependence of the best algorithm on the standard used to transmit the OFDM signal, there are many opportunities to expand upon this research.

First, the analysis and performance of the algorithms for carrier phase offset tracking, OFDM symbol timing, and data symbol timing could be performed and simulated using System View.

Second, the analysis performed in Chapter IV could be expanded from the AWGN channel to include a mulipath channel. This significantly increases the difficulty of the analysis, but would provide a useful comparison with the actual environment expected to be present when using OFDM systems.

Third, the algorithms described in Chapter IV that are more suited to continuous transmission systems could be analyzed and simulated with System View. The cyclic prefix correlation and the virtual carrier algorithms may provide interesting results.

Fourth, the System View designs implemented in Chapter V could be implemented with an FPGA card and compared to current receiver cards available commercially. System View offers an HDL design studio to convert a System View circuit design into synthesizable VHDL or Verilog source code. The FPGA lab in the Cryptologic Research Lab at the Naval Postgraduate School has the required software and hardware to use the VHDL source code to program an FPGA card.

Finally, the physical layer designs implemented could be used as a starting point for a higher layer, software implemented analysis of the received OFDM signals.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A

The file used as the input to the QAM mapper token is shown below. No header was provided, and line one of the file starts with $n = 16$, as shown below. It follows the exact format provided in the 16QAM notepad file, 16QAM.txt, provided in the Commlib folder in the System View Example folder. This usually can be found in the C drive under System View folder in the Program Files folder.

n=16

Symbol	I	Q
0	-3.0	-3.0
1	-3.0	-1.0
2	-3.0	3.0
3	-3.0	1.0
4	-1.0	-3.0
5	-1.0	-1.0
6	-1.0	3.0
7	-1.0	1.0
8	3.0	-3.0
9	3.0	-1.0
10	3.0	3.0
11	3.0	1.0
12	1.0	-3.0
13	1.0	-1.0
14	1.0	3.0
15	1.0	1.0

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

This worksheet from Mathcad 2001 shows the calculation of the probability of detection and the probability of false alarm for the Ricean and Rayleigh distributions used in the sliding-window correlation detector in Section 1 of Chapter V.

$r := 0, 0.001 .. 1.6$ $s := \sqrt{2 \cdot (0.1015625)^2}$ $\sigma := \sqrt{1} \cdot \sqrt{0.1015625}$ $\eta := 0.1538905$

$$P(r,\sigma) := \frac{r}{\sigma^2} \cdot e^{\frac{-\left(r^2+s^2\right)}{2\cdot\sigma^2}} \cdot I0\left(r \cdot \frac{s}{\sigma^2}\right)$$

$s = 0.1436310649$

$$P_D(r,\sigma) := \int_{\eta}^{2} \frac{r}{\sigma^2} \cdot e^{\frac{-\left(r^2+s^2\right)}{2\cdot\sigma^2}} \cdot I0\left(r \cdot \frac{s}{\sigma^2}\right) dr$$

$$P_{FA}(\eta,\sigma) := e^{\frac{-\eta^2}{2\cdot\sigma^2}}$$

$P_{FA}(\eta,\sigma) = 0.8899502509$

$$P_G(r) := \frac{1}{\sqrt{2\cdot\pi}} \cdot e^{\frac{-(r-s)^2}{2}}$$

$$P_R(r,\sigma) := \frac{r}{\sigma^2} \cdot e^{\frac{-r^2}{2\cdot\sigma^2}}$$

$P_D(r,\sigma)$
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9
0.9

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Robert W. Chang, "Synthesis of Band-Limited Orthogonal Signals for Multi-channel Data Transmission," *The Bell System Technical Journal*, Vol. XLV, No. 10, pp. 1775-1796, December 1966.

2. S. B. Weinstein and Paul M. Ebert, "Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier Transform," *IEEE Transactions on Communication Technology*, Vol. Com-19, No. 5, pp. 628-634, October 1971.

3. Juha Heiskala and John Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*, Sams Publishing, Indianapolis, Indiana, 2002.

4. Richard van Nee and Ramjee Prasad, *OFDM for Wireless Multimedia Communications*, Artech House, Boston, Massachusetts, 2000.

5. Patrick A. Count, "Performance Analysis of OFDM in Frequency-Selective, Slowly Fading Nakagami channels," Master's Thesis, Naval Postgraduate School, Monterey, California, 2001.

6. Donovan I. Oubre, "Capabilities and Limitations of Orthogonal Frequency-Division Multiplexing in Wireless Applications," Master's Thesis, Naval Postgraduate School, Monterey, California, 2001.

7. Chi-han Kao, "Performance of the IEEE 802.11a Wireless LAN standard over Frequency-Selective, Slow, Ricean Fading Channels," Master's Thesis, Naval Postgraduate School, Monterey, California, 2002.

8. Institute of Electrical and Electronics Engineers, 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 18 March 1999.

9. Institute of Electrical and Electronics Engineers, 802.11b, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, 16 September 1999.

10. Institute of Electrical and Electronics Engineers, 802.11a, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer Extension in the 5 GHz Band*, 16 September 1999.

11. Institute of Electrical and Electronics Engineers, 802.11g, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*, 12 June 2003.

12. Institute of Electrical and Electronics Engineers, 802, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, 08 March 2002.

13. Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage*, pp.264-265, Prentice Hall, Upper Saddle River, New Jersey, 1995.

14. William Carney, "IEEE 802.11g New Draft Standard Clarifies Future of Wireless LAN," White paper, Texas Instruments Incorporated, 11 January 2002, http://focus.ti.com/pdfs/vf/bband/802.11g_whitepaper.pdf last accessed 29 January 2004.

15. Maurice L. Schiff, "802.11a System Simulation Using System View by Elanix," AN 140, Elanix Incorporated, Westlake Village California, November 2002, http://www.elanix.com/pdf/an140.pdf last accessed 29 January 2004.

16. Ralph D. Hippenstiel, *Detection Theory: Applications and Digital Signal Processing*, pp. 63-159, CRC Press, Boca Raton, Florida, 33431,2002.

17. Timothy M. Schmidt and Donald C. Cox, "Robust Frequency and Timing Synchronization for OFDM," *IEEE Transactions on Communications*, Vol. 45, No. 12, pp. 1613-1621, December 1997.

18. Meng_Han Hsieh and Che-Ho Wei, "A Low-Complexity Frame Synchronization and Frequency Offset Compensation Scheme for OFDM Systems over Fading Channels," *IEEE Transactions on Vehicular Technology*, Vol. 48, No. 5, pp.1596-1609, September 1999.

19. Flavio Daffara and Antoine Chouly, "Maximum Likelihood Frequency Detectors for Orthogonal Multicarrier Systems," *Proc. of IEEE Trans. On Communications*, Geneva, Switzerland, pp. 766-771, May 1993.

20. Marco Luise and Ruggero Reggiannini, "Carrier Frequency Acquisition and Tracking for OFDM Systems," *IEEE Transactions on Communications*, Vol. 44, no. 11, pp.1590-1598, November 1996.

21. Flavio Daffara and Ottavio Adami, "A New Frequency Detector for Orthogonal Multicarrier Transmission Techniques," *Proc. of IEEE Veh. Technol. Conf.*, Chicago, IL, pp.804-809, July 1995.

22. Minoru Okada, Shinsuke Hara, Shozo Komaki and Norihiko Morinaga, "Optimum Synchronization of Orthogonal Multi-Carrier Modulated Signals," *Proc. IEEE PIMRC*, Taipei, Taiwan, pp. 863-867, October 1996.

23. Paul H. Moose, "A Technique for Orthogonal Frequency Division Multiplexing Frequency Offset Correction," *IEEE Transactions on Communications*, Vol. 42, No. 10, pp. 2908-2914, October 1994.

24. Sergio Barbarossa, Massimiliano Pompili, and Georgios B. Giiannakis, "Channel-Independent Synchronization of Orthogonal Frequency Division Multiple Access Systems," *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 2, pp. 474-486, February 2002.

25. Michael Speth, Stefan Fechtel, Gunnar Fock, and Heinrich Meyr, “Optimum Receiver Design for OFDM-Based Broadband Transmission-Part II,” *IEEE Transactions on Communcations,* Vol.49, No. 4, pp. 571-578, April 2001.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Chairman, Code EC
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA

4. Professor Frank Kragh Code EC/Kh
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA

5. Professor R. Clark Robertson Code EC/Rc
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA

6. Professor Ronald Pieper
   Department of Electrical Engineering
   University of Texas
   Tyler, TX

7. Director, National Security Agency
   ATTN: Robert Eubank
   Fort Meade, MD

8. Nick Triska
   SAIC
   Monroe, CT

9. Nathan Beltz
   Department of Electrical Engineering
   Naval Postgraduate School
   Monterey, CA

10. Commanding Officer, Space and Naval Warfare Systems Center
    ATTN: Mr. Dale Bryan Code 2371
    San Diego, CA

11. Commanding Officer, Space and Naval Warfare Systems Center
ATTN: Dr. Rich North Code 2846
San Diego, CA

12. LCDR Keith Lowham
Chief of Naval Operations, Code N77
Arlington, VA